Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

PLATAFORMA WEB PARA EL DISEÑO Y EJECUCIÓN DE MODELOS DE APRENDIZAJE PROFUNDO

Alumno: David Valdivia Vico

Tutor: Prof. D. Antonio Jesús Rivera Rivas
Dpto: Departamento de Informática

Tutora: Prof. Dña. María Dolores Pérez Godoy
Dpto: Departamento de Informática

Junio, 2022

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Antonio Jesús Rivera Rivas y Doña María Dolores Pérez Godoy, tutores del Proyecto Fin de Carrera titulado: Plataforma Web para el diseño y ejecución de modelos de aprendizaje profundo, que presenta David Valdivia Vico, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, JUNIO de 2022

El alumno:

VALDIVIA VICO DAVID - 77376145M

Firmado digitalmente por VALDIVIA VICO DAVID - 77376145M
Fecha: 2022.07.02 10:40:41 +02'00'

David Valdivia Vico

Los tutores:

RIVERA RIVAS ANTONIO JESUS - 26015863B

Firmado digitalmente por RIVERA RIVAS ANTONIO JESUS - 26015863B
Fecha: 2022.06.30 12:44:46 +02'00'

Antonio Jesús Rivera Rivas

PEREZ GODOY MARIA DOLORES - 75093149N

Firmado digitalmente por PEREZ GODOY MARIA DOLORES - 75093149N
Fecha: 2022.06.30 13:22:02 +02'00'

María Dolores Pérez Godoy

Índice

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN Y MOTIVACIÓN.

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos inspirados en los criterios y recomendaciones que establece la norma UNE 157801:2007 - “Criterios Generales para la elaboración de proyectos de Sistemas de información”.

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 0 (Definiciones y abreviaturas).

1.1. Introducción.

Vivimos en la época del Big Data donde estamos rodeados de datos por todos lados, los datos nos pueden dar una gran cantidad de información útil o de información inútil, todo depende de cómo tratamos y mostramos estos datos.

Una forma de tratar tal cantidad de datos es mediante el aprendizaje automático, una disciplina de la inteligencia artificial. El aprendizaje automático consiste en crear programas capaces de aprender conforme a su experiencia (Russell & Norvig, 2004). Los algoritmos de aprendizaje automático trabajan de forma iterativa, en la primera iteración, la experiencia es nula, por consiguiente, el rendimiento es muy bajo. A medida que se incrementa el número de iteraciones el rendimiento mejora.

En el aprendizaje automático nos encontramos con las redes neuronales (IBM, 2021). Las redes neuronales son modelos simplificados que emulan el modo en el que el cerebro humano procesa la información. Este modelo está formado por las neuronas que son las unidades más básicas y estas neuronas son organizadas en capas.

Las redes neuronales normalmente están formadas por tres partes o conjuntos de capas. La primera capa es una capa de entrada, donde se representan los campos de entrada, el segundo conjunto de capas se denomina capas ocultas y por último tenemos la capa de salida con una unidad o unidades que representa el campo que buscábamos como resultado.

Para que las redes neuronales nos arrojen una solución con sentido las muestras del conjunto de entrenamiento deben ir pasándose de forma iterativa a la red neuronal que actualizará sus parámetros según un algoritmo de entrenamiento, de esta forma y poco a poco irán mejorando sus resultados. En las primeras iteraciones todo es aleatorio, pero a medida que van pasando las ejecuciones comienza a relacionar entre sí los datos de entrada y va mostrando predicciones cada vez más correctas.

En la actualidad las redes de aprendizaje profundo o Deep Learning son utilizadas en muchas y variadas áreas, desde el deporte para el análisis de estadística y predicción de resultados hasta la medicina para el diagnóstico temprano de ciertas enfermedades. Uno de los mayores usos de estas redes está relacionado con la visión artificial, permitiéndonos, por ejemplo, reconocer personas en tiempo real a través de un vídeo.

El Deep learning (Goodfellow, Bengio, & Courville, 2016) parte de comenzar la tarea a partir de conceptos simples, que vayan relacionándose entre sí, y con esto construir una jerarquía de conceptos más complejos a partir de los más simples.

En este tipo de modelos se incentiva que los ordenadores extraigan su propio conocimiento y desarrollen su propio esquema de razonamiento evitando usar el conocimiento que el humano ha introducido en la red.

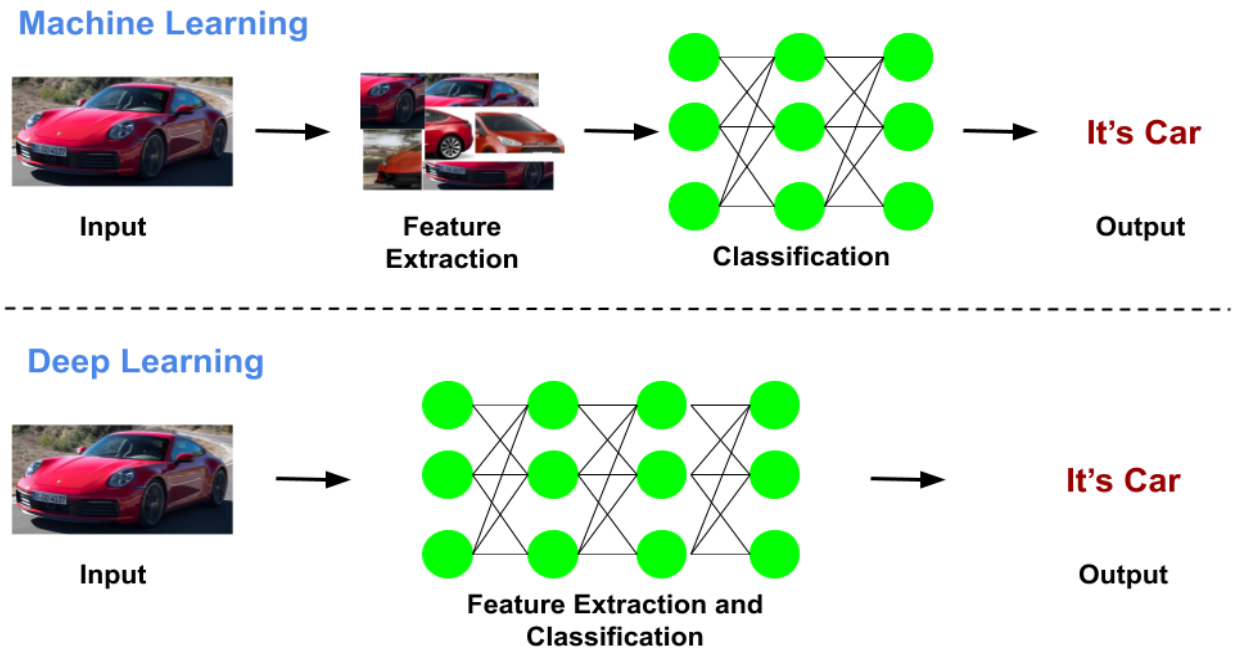

Figura 1: Comparativa entre aprendizaje poco profundo y Deep learning. Fuente.

En la Figura 1 se puede observar dos tipos de redes, una de aprendizaje poco profundo (parte superior) y una de Deep learning (parte inferior), si bien ambas arrojan el mismo resultado, vemos como la red de arriba necesita una extracción de características, normalmente realizada por el desarrollador, para que la red pueda relacionar y clasificar la imagen correctamente. Sin embargo, la red de Deep learning hace todo esto sola y finaliza clasificando la imagen correctamente.

El objetivo de este proyecto es realizar una aplicación web divulgativa donde el cliente pueda ejecutar y modificar diferentes tipos de redes, desde simples a complejas, para que el usuario aprenda las diferentes aplicaciones que tienen cada tipo de red y cuáles son las principales características de cada una de ellas.

1.2. Motivación.

Como se indicaba en la introducción de este apartado, la inteligencia artificial es un campo de la informática que está en auge en las últimas décadas, por consiguiente, la principal motivación de este proyecto es la de ayudar a la gente a adentrarse en esta rama, más concretamente en la sección de redes neuronales. La aplicación web se ha desarrollado pensando en el usuario que tiene pocos o ningún conocimiento en inteligencia artificial y que, a partir de usar la web, descubre el potencial del aprendizaje automático, cree una curiosidad acerca de este campo y se inicie a aprender, porque el conocimiento que no se comparte pierde valor. Específicamente, la aplicación va a permitir al usuario el diseño, el entrenamiento y la ejecución de distintos tipos de redes neuronales.

Por otro lado, no es habitual encontrarse con aplicaciones web que te permiten ejecutar modelos de aprendizaje automático en la parte del cliente, lo cual abre un abanico de posibilidades respecto las aplicaciones que se ejecutan en la parte del servidor.

1.3. Objetivos del trabajo.

El desarrollo del proyecto, Plataforma Web para el diseño y ejecución de modelos de aprendizaje profundo, se ha seguido teniendo en cuenta el cumplimiento de los siguientes objetivos.

- Implementación de un módulo para el diseño de modelos de aprendizaje profundo que permita editar o modificar el código de los modelos.
- Diseño de un módulo para el aprendizaje de modelos donde el usuario podrá configurar los parámetros de estos y entrenarlos.
- Desarrollo de un módulo para la ejecución de los modelos ya entrenados en la máquina cliente.

1.4. Estructura de la memoria.

La presente memoria se estructura de la siguiente manera:

- **Primer capítulo: Introducción y motivación.** En este capítulo se introduce el campo del aprendizaje automático y se explica de forma superficial el enfoque que se ha dado para facilitar el aprendizaje del aprendizaje automático.
- **Segundo capítulo: Contexto del trabajo.** Este capítulo explica la planificación y desarrollo de la solución aplicada. Para ello se usan distintas técnicas de Ingeniería del software.
- **Tercer capítulo: Análisis y planificación.** El objetivo de este capítulo es descomponer los requisitos para llevar a cabo el proyecto.
- **Cuarto capítulo: Diseño.** En este capítulo se explica el estudio que se ha realizado para hacer el diseño de esta aplicación.
- **Quinto capítulo: Desarrollo del proyecto.** El objetivo de este capítulo es descomponer los requisitos para llevar a cabo el proyecto.
- **Capítulos finales:** Apéndices, manuales, definiciones y abreviaturas.

2. CONTEXTO DEL TRABAJO.

En esta sección se va a hacer una introducción al aprendizaje automático, Deep learning, tensorflow, tecnologías usadas en el proyecto y metodologías de ingeniería del software.

2.1. Aprendizaje automático.

Hay diferentes tipos de aprendizaje automático, esto son: el supervisado, el no supervisado y por refuerzo. Los algoritmos de aprendizaje no supervisado son el paradigma que consigue producir conocimiento únicamente de los datos que se proporcionan como entrada sin necesidad en ningún momento de explicarle al sistema que resultado queremos obtener. El objetivo del aprendizaje no supervisado es modelar la estructura o distribución subyacente en los datos para aprender más sobre los mismos. En cambio, en el aprendizaje supervisado se basa en descubrir la relación existente entre unas variables de entrada y unas variables de salida, es decir, mostrarle al sistema el resultado que deseas obtener para un determinado valor de entrada. Tras mostrarle muchos ejemplos (si se dan las condiciones) el algoritmo será capaz de dar un resultado correcto incluso cuando le muestras valores que no había visto antes.

Una técnica del aprendizaje automático muy usado son las redes neuronales, estas redes necesitan una gran cantidad de datos etiquetados para funcionar correctamente.

El algoritmo de aprendizaje automático supervisado necesita que le enseñemos la respuesta correcta y a partir de la misma es capaz de aprender de sus errores. En las primeras iteraciones del algoritmo la tasa de error es muy alta ya que prácticamente las salidas que muestra son aleatorias, pero a medida que va “aprendiendo” se reduce la tasa de error.

El proceso del aprendizaje automático es el siguiente:

1. Obtener o recopilar datos con la solución.
2. Crear un modelo predictivo.

3. Entrenar el modelo con los datos obtenidos en el punto 1.
4. Probar el modelo con datos nuevos.

Clásicamente en el aprendizaje supervisado destacan dos subcategorías, la clasificación y la regresión.

La clasificación consiste en predecir a que conjunto de categoría pertenece un/os datos de entrada, basándose en observaciones pasadas. Los dos tipos de clasificación más conocidos son: clasificación binaria donde solo existen dos conjuntos o posibles soluciones y clasificación multiclase en el que existen varios conjuntos o posibles soluciones.

El objetivo del análisis de regresión (España, 2020) es establecer un método para la relación entre un cierto número de características y una variable objetivo continua. Nos puede ayudar a predecir la edad de una persona o calcular el precio de una vivienda.

El aprendizaje supervisado ha sido el paradigma que más aplicaciones prácticas ha tenido en las últimas décadas liderando la corriente al alza que ha sufrido el campo de la inteligencia artificial, ya que nos puede ser útil en cualquier campo. Si existe alguna relación entre los datos entrada, el algoritmo encontrará la relación entre ellos.

El aprendizaje por refuerzo debe aprender cómo se comporta el entorno mediante recompensas (refuerzos) o castigos, derivándose del éxito o fracaso respectivamente.

También es importante hablar del campo de la visión artificial. Es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por una máquina. Es como dotar a la máquina con ojos para que puede ver. Entre sus aplicaciones más utilizadas están el reconocimiento de objetos, detección de sucesos o tratamiento de imágenes. Si combinamos la visión artificial (generador de información) y el aprendizaje automático (procesador de la información) podemos construir multitud de aplicaciones. Por ejemplo, podemos crear un modelo de clasificación multiclase de números manuscritos desde el 0 al 9 (Figura 2). La entrada es una imagen tratada por la visión artificial para transformarla en un lenguaje que la máquina sea capaz de

entender y la salida es la predicción que la máquina hace del número que cree que es.

Figura 2: Pequeña muestra del dataset MNIST para la clasificación de números. Fuente

Una red neuronal es un modelo computacional basado en un gran conjunto de unidades neuronales simples de forma aproximadamente análoga al comportamiento observado en los axones de las neuronas de un cerebro biológico (Schmidhuber, 2015)

Como observamos en la Figura 3 cada uno de los nodos representa una neurona y cada flecha representa una conexión entre la salida de una neurona y la entrada de la otra. El valor de salida de cada neurona es multiplicado por un valor de peso. Cada neurona puede tener un valor del peso diferente ya que van buscando reducir el valor obtenido por la función de pérdida.

Una función de pérdida es una función que relaciona un evento con un valor numérico representando el coste a nivel económico del evento.

Las neuronas también se pueden ver afectadas por la función de activación, una función umbral que modifica el valor de la neurona antes de pasar a otra neurona.

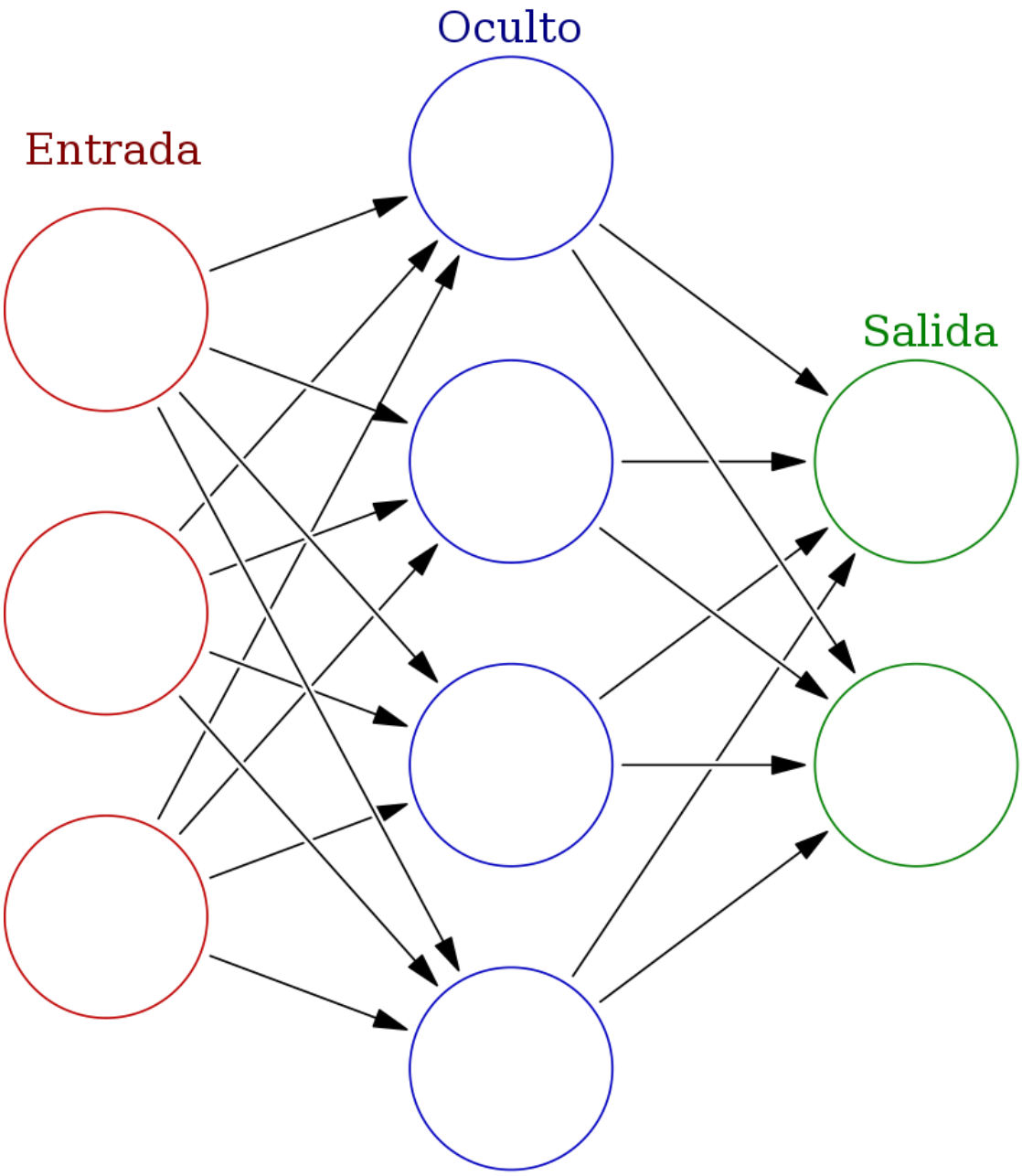

Figura 3: Representación gráfica de una red neuronal. Fuente

Es importante también el concepto de aprendizaje profundo. Nos referimos a aprendizaje profundo (en inglés, Deep Learning) (Torres, 2020) cuando el modelo basado en redes neuronales está compuesto por múltiples capas ocultas. Como se ha comentado antes estas redes por así decirlo son independientes, no necesitan que el usuario le descomponga todos los datos de entrada para que la red los entienda y

realice una clasificación correcta. Observando la Figura 4 vemos cómo se ven claramente tres grupos de capas de neuronas. El primer grupo de color azul se corresponde con las neuronas de entrada las cuales ingesta los datos de entrada, el segundo conjunto de neuronas de color naranja es la parte donde la red realiza y calcula todas las relaciones de los datos que se han introducido y por último la capa final de color verde que nos arroja la predicción que ha realizado la red.

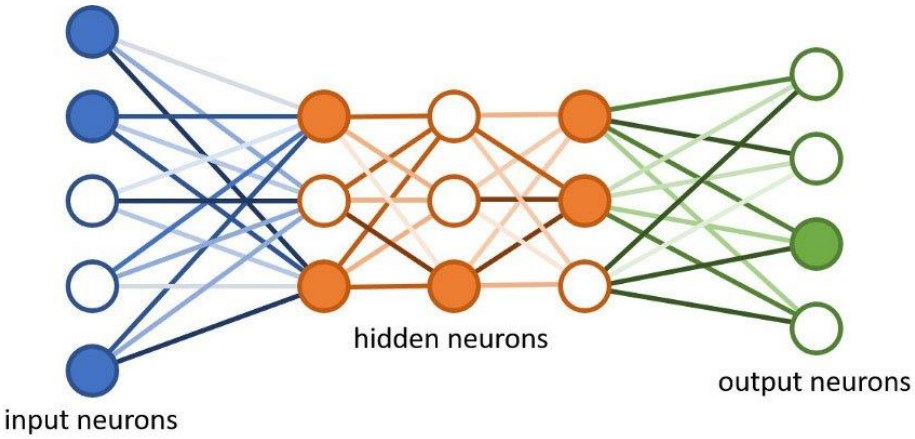

Figura 4: Esquema de una red de aprendizaje profundo. Fuente

Las redes neuronales clásicas y el Deep learning tratan de simular el comportamiento del cerebro humano, pero las redes neuronales clásicas necesitan un mayor procesamiento de los datos para que la red pueda aislar correctamente las características de cada dato y poder así, crear relaciones entre las diferentes características. Sin embargo, las redes de Deep learning no necesitan este procesamiento adicional ya que la propia red se encarga de separar todas las características de forma que puede detectar relaciones en los datos que nosotros los humanos, no podemos detectar a simple vista.

2.2. Herramientas

2.2.1. Elección framework aprendizaje profundo

Hasta hace unos pocos años para generar modelos de aprendizaje automático se debían de programar en la parte del servidor usando lenguajes como Python[1] por lo que para poder realizar una web que ejecutase modelos de aprendizaje automático necesitabas un back-end que era el que realmente creaba, entrenaba y ejecutaba los modelos.

Sin embargo, han aparecido diferentes librerías JavaScript[2] que nos permiten crear, entrenar y ejecutar estos modelos directamente desde la web, asumiendo el coste computacional el cliente (usuario final).

Entre las librerías más utilizadas se encuentran ConvNetJS[3] (Figura 6) y Tensorflow.js[4] (Figura 5). Ambas librerías funcionan de forma muy similar y son una excelente opción para la realización del proyecto ya que nos permiten entrenar redes neuronales desde el navegador (cliente).

Figura 5: Logo de Tensorflow.js. Fuente

[1] https://www.python.org/
[2] https://developer.mozilla.org/es/docs/Web/JavaScript
[3] https://cs.stanford.edu/people/karpathy/convnetjs/
[4] https://www.tensorflow.org/

Figura 6: Logo de ConvNetJS. Fuente

Para este proyecto hemos decidido usar la librería Tensorflow.js debido a su compatibilidad con la librería de Python Tensorflow. Gracias a esta compatibilidad seremos capaces de ejecutar modelos creados desde Tensorflow (servidor), además está muy bien documentada y se ha construido una gran comunidad en torno a esta librería.

2.2.2. Sistema operativo y lenguaje de programación.

Al usar la librería de JavaScript es recomendable escoger un framework basado en JavaScript para la realización de la interfaz gráfica (front-end).

Hay bastantes frameworks basados en JavaScript, pero los más utilizados actualmente son los que podemos ver en la Figura 7.

React.js[5] es una biblioteca de JavaScript para crear interfaces de usuario. Fue lanzada en 2013 y creado por Jordan Walke, un ingeniero de software de Facebook[6]. La primera implementación de este framework fue optimizar la carga de contenidos en la red social, poco a poco la librería empezó a mejorar hasta la actualidad.

[5] https://es.reactjs.org/
[6] https://www.facebook.com/

AngularJS[7] es un framework MVC (Modelo Vista Controlador), desarrollado por Google[8] en 2009 para el desarrollo web front-end que permite crear aplicaciones SPA (Single-Page Applications).

Vue.js[9] es un framework progresivo para construir interfaces de usuario. Esta librería solo está enfocada a la capa de visualización.

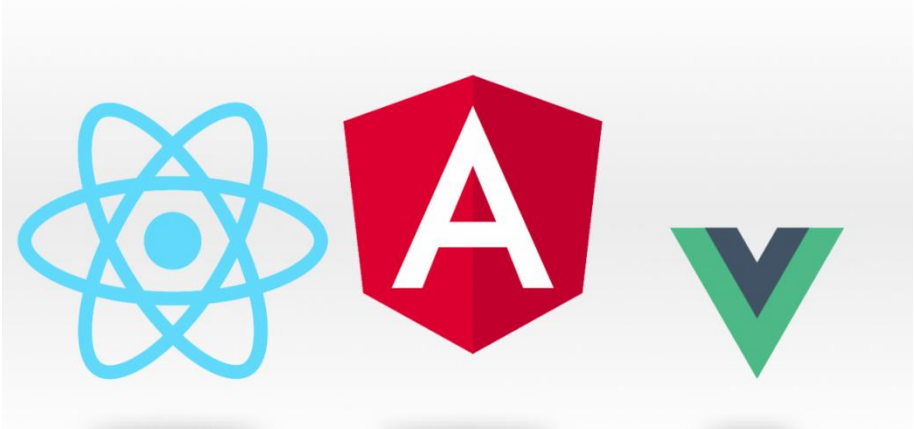

Figura 7: De izquierda a derecha React.js, Angular.js y Vue.js. Fuente

Son tres frameworks muy potentes y el proyecto se puede realizar con cualquiera de los tres, debido a que anteriormente había tenido una iniciación con la librería React.js se ha decido desarrollar el software de este proyecto con esta librería.

Para el desarrollo del proyecto se hará uso de dos equipos como se especifica en el apartado 3.2.1, una torre montada por piezas con sistema operativo Windows 10 Pro y un equipo portátil de 2016 con sistema operativo Windows 11 (Figura 8). Para poder trabajar con ambos equipos el código se va a almacenar en un repositorio remoto de forma que sea accesible desde cualquier lugar con una conexión a internet.

[7] https://angularjs.org/
[8] https://google.es/
[9] https://vuejs.org/

Figura 8: Logotipos del sistema operativo Windows en sus versiones 10 y 11. Fuente

2.2.3. Gestión del código.

El código se va a mantener mediante un sistema de control de versiones Git[10] (Figura 9), universalmente conocido y utilizado por la mayoría de los desarrolladores, a la vez se va a hacer uso de los repositorios de la compañía GitHub[11] (Figura 10) para poder trabajar en paralelo desde diferentes equipos.

Figura 9: Logotipo de Git. Fuente

[10] https://git-scm.com/
[11] https://github.com/

Figura 10: Logotipo de GitHub. Fuente

2.2.4. Entorno de desarrollo.

El desarrollo y depuración del software se han realizado empleando el IDE (Integrated Development Environment) Visual Studio Code[12] (Figura 11) de Microsoft. Se trata de un software gratuito y de código abierto. Se puede usar con la gran mayoría de lenguajes de programación actuales.

Figura 11: Logotipo de Visual Studio Code. Fuente

[12] https://code.visualstudio.com/

2.3. Aprendizaje automático en Tensorflow.js

El objetivo de este punto es explicar a nivel de programación cómo crear, entrenar y evaluar modelos de aprendizaje automático usando la librería de Tensorflow.js. Esto nos permitirá comprender las facilidades y complejidades que tiene la librería a la hora de realizar un proyecto como este.

En el apartado anterior se han comentado las diferentes opciones que tenemos a nivel software para realizar modelos de aprendizaje automático en una aplicación web. Partimos de la premisa de que todo el software está instalado correctamente y nos disponemos a realizar la aplicación.

Esta explicación está enfocada a Tensorflow.js que es la librería correspondiente a JavaScript, pero la forma de trabajo es muy similar a la librería principal de Tensorflow desarrollada para el lenguaje Python.

Para poder crear y entrenar modelos en Tensorflow.js necesitamos 4 premisas:

- Dataset: datos de entrenamiento y evaluación
- Crear arquitectura
- Entrenar arquitectura
- Evaluar arquitectura

2.3.1. Dataset.

Vivimos en una época en la que tenemos a la altura de una búsqueda en la web gran cantidad de información. Para realizar modelos de aprendizaje automático necesitamos una gran cantidad de información que podemos obtener por ejemplo de internet, pero esta información debe de ser tratada antes para que se adapte correctamente a lo que el modelo necesita.

Dependiendo de del tipo de modelo que queramos realizar, los datos de entrenamiento y evaluación serán diferentes. Por ejemplo, si queremos hacer un modelo que sea capaz de reconocer números escritos a mano, los datos de entrada serán una serie de imágenes con números dibujados a mano, pero ¿valen imágenes

de todas las resoluciones? ¿Las imágenes deben de estar en blanco y negro o a color? ¿Hay que realizar algún tratamiento a la imagen? De igual forma pasa si nuestra red solamente se entrena con vectores numéricos, hay que detectar correctamente cuantas entradas y salidas necesitamos, si los datos son enteros, flotantes …

Una vez que hemos analizado y tratado los datos necesitamos convertirlos a tensores para que el sistema pueda interpretar los datos y sacar relaciones de los mismos.

Un tensor es una estructura de datos de n-dimensiones que permite representar todos los tipos de datos. Los tensores tienen tres propiedades, nombre, forma y tipo de dato.

Los tensores son creados a partir de la relación de los datos de entrada junto con su categoría, estos tensores nos permitirán entrenar la futura red que creemos.

2.3.2. Crear arquitectura.

Una vez tenemos preparados los datos, pasamos a la creación del modelo. Es importante diferenciar entre modelo y arquitectura. Una arquitectura es la definición de la estructura que va a tener un modelo. El número de capas que tiene, cuantas entradas y salidas en cada capa, qué tipo de capa son cada una, etc. Sin embargo, un modelo es el conjunto de la arquitectura con todos los pesos asignados a cada uno de los nodos de las diferentes capas. Un modelo ya ha sido entrenado y aunque se pueda volver a entrenar, es completamente funcional y evaluable. La arquitectura forma parte de un modelo.

A la hora de hacer la arquitectura de un modelo de aprendizaje automático tenemos algunas directrices por las que guiarnos, pero en general puedes crear la red de multitud de formas. No hay una arquitectura exacta para cada ejemplo, todas valen, pero algunas son más óptimas que otras y esto es un poco lo interesante, realizar un modelo y poco a poco ir mejorándolo analizando la red que has hecho y como la has hecho.

Existen páginas que nos muestran algunas arquitecturas que funcionan realmente bien en diferentes tipos de problemas[13].

Si ponemos como ejemplo la red de la Figura 4 los nodos de la red son cada uno de los círculos. Las capas de la red están formadas por diferentes nodos, en este caso las capas se pueden visualizar fácilmente en columnas.

Para comenzar a crear el modelo tenemos dos opciones: tf.sequential y tf.model. Tf.sequential es una pila lineal de capas mientras que tf.model permite crear un gráfico de capas siempre y cuando no tengan ciclos (Google, 2021). Dependiendo de la solución que necesitamos eligiéremos un tipo u otro, por regla general el tipo sequential resuelve mejor, problemas más simples. Una vez hemos decidido el tipo de modelo a crear tenemos que empezar a agregar las capas que van a formar la red. Aquí se abre un gran abanico de posibilidades.

Una de las capas más importantes es la primera capa ya que definen la forma de los tensores de entrada. Para añadir nuestra primera capa al modelo nos ayudaremos de la función model.add() y por parámetro le pasaremos el tipo de capa que deseamos. Hay gran multitud de tipos de capas, todo depende de la solución que deseamos aplicar, algunos ejemplos de estas capas son: Dense, Convolutional, Merge, Normalization… .

Simplemente elegimos la red que deseamos crear y le pasamos algunos parámetros que necesitan como son el número de neuronas de cada capa, la función de activación que va a usar y la forma. Esta forma es la que más tarde se le pedirá al tensor de entrada, si no comparten forma el modelo no podrá interpretar los datos.

También es importante definir bien la última capa o capa de salida ya que si, por ejemplo, estamos con un problema de clasificación perros y gatos, las posibles soluciones son: perro, gato o desconocido, por lo que la capa de salida nos debe de dar un porcentaje de cada solución.

[13] https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/

Normalmente la última capa tiene la función de activación sigmoid la cual transforma la predicción de la red neuronal al rango 0-1 permitiéndonos ver de forma más sencilla la predicción del modelo.

```
const model = tf.sequential();
const learningRate = 0.01;
const numberOfEpoch = 40;
const optimizer = tf.train.adam(learningRate);


model.add(
  tf.layers.dense({
    units: 10,
    activation: "softmax",
    inputShape: [xTrain.shape[1]],
  })
);

model.add(tf.layers.dense({ units: 3, activation: "sigmoid" }));

model.compile({
  optimizer: optimizer,
  loss: "categoricalCrossentropy",
  metrics: ["accuracy"],
});
```

Figura 12 : Fragmento de código usado para realizar un modelo de aprendizaje automático sencillo

En la Figura 12 se puede observar cómo se crea un modelo muy básico. Primero definimos que tipo de modelo va a ser y algunos parámetros como la tasa de aprendizaje, el número de iteraciones, el optimizador que se va a usar. Después se comienza a agregar capas al modelo, en este caso la primera capa que se añade es del tipo “dense”, con 10 neuronas y una función de activación “softmax”. Tras agregar todas las capas al modelo se llama a la función “compile” la cuál recibe como parámetros el optimizador anteriormente definido, la función de pérdida y la función métrica.

2.3.3. Entrenar arquitectura.

Antes de comentar cómo se entrena una arquitectura se va a explicar todos los parámetros que se utilizan en este proceso de forma que se globalice una idea de cómo trabaja la librería de Tensorflow.js.

- Número de iteraciones: número de iteraciones que se van a realizar durante el entrenamiento. A mayor número de iteraciones más tiempo de ejecución, pero también va mejorando la predicción.
- Tasa de entrenamiento: valor entre 0 y 100. Es la división en porcentaje de los datos que se dedican a entrenar y los que se dedican a evaluar.
- Optimizador: es una función que nos acerca a la solución óptima.
- Función de perdida: evalúa la desviación entre diferentes predicciones del modelo. Mientras más bajo sea este valor, más eficiente es la red.
- Métrica: es una función que permite hacer un seguimiento del rendimiento de la red.

Una vez hemos acabado de agregar capas a nuestro modelo, aplicamos la función model.compile() la cual recibe como parámetros una función de optimización, una función de pérdida y unas métricas.

Para entrenar el modelo aplicamos la función model.fit() (Figura 13). Esta función recibe por parámetros los datos de entrenamiento, el número de iteraciones que va a realizar y los datos de validación. Los datos de validación son los datos que se destinan a la validación del modelo durante el entrenamiento, procedentes del dataset. Adicionalmente se le pueden agregar algunos parámetros para que nos muestre una gráfica de cómo avanza el entrenamiento.

```
const history = await model.fit(xTrain, yTrain, {
  epochs: numberOfEpoch,
  validationData: [xTest, yTest],
  callbacks: {
    onEpochEnd: async (epoch, logs) => {
      if(verbose)
      document.getElementById("Result").innerHTML+="<p>Epoch: " + epoch + " Logs:" + logs.loss+"</p>";
      await tf.nextFrame();
    },
  },
});
return model;
```

Figura 13: Fragmento de código correspondiente al entrenamiento de un modelo.

Tras este proceso tendremos un modelo completamente funcional el cual se puede evaluar, exportar o volver a entrenar (reentrenar).

2.3.4. Evaluar modelo.

Evaluar el modelo por parte del usuario es muy sencillo, simplemente se deben de preparar unos datos de entrada introducidos por el usuario y “traducirlos” para que formen un tensor y este se envíe al modelo.

El tratamiento de los datos es muy similar al realizado para crear los datos de entrenamiento. Una vez hemos realizado este tratamiento de datos se llama a model.evaluate(), pasándole los datos de entrada como parámetro, para obtener una predicción del modelo. Si es un problema de clasificación el modelo nos devolverá un vector de tamaño n siendo n el número de posibles soluciones. El valor más alto de este vector será la predicción del modelo. Cabe destacar que la suma de todos los valores del vector da 1.

2.3.5. Parámetros típicos de Tensorflow.js

A continuación, se describen diferentes parámetros que se pueden configurar a la hora de crear, entrenar y evaluar una red neuronal.

2.3.5.1. Tipos de capas.

Como se ha visto en el apartado 2.3.2 tenemos diferentes tipos de capas a la hora de crear un modelo. En la Tabla 1 podemos ver un resumen de los tipos más utilizados.

Nombre	Descripción
Dense	Esta función se utiliza para crear capas completamente conectadas, en las que cada salida depende de cada entrada.
Convolutional	Existen tres tipos de capas Convolutional, 1d, 2d y 3d. Estas capas nos permiten crear un núcleo de convolución que se convoluciona con los datos de entrada sobre el número de dimensiones elegido según la capa.
Merge	Se trata de un conjunto de funciones que definen diferentes operaciones como añadir o concatenar tensores a una capa.
Normalization	Nos permite normalizar la activación de la capa anterior es decir mantiene la activación media cerca de 0 y la desviación estándar de activación cerca de 1.

Tabla 1: Descripción de los tipos de capas más utilizados.

2.3.5.2. Funciones de activación.

Hay diferentes funciones de activación, las más utilizadas son softmax, sigmoid y ReLU. En la Tabla 2 se puede ver una breve descripción de cada una.

Nombre	Descripción	Características
Softmax	Transforma las salidas a una representación en forma de probabilidades, de forma que la suma de todas las probabilidades da 1	• Acotada entre 0 y 1 • Buena en capas finales • Utilizada para normalizar tipo multiclase
Sigmoid	Transforma los valores de entrada donde los más altos tienden a 1 y los valores más bajos tienden a 0	• Acotada entre 0 y 1 • Lenta convergencia • Buena en capas finales
ReLU	Transforma los valores de entrada dejando a cero los valores negativos y dejando tal como estaban los datos positivos	• No está acotada • Buen desempeño con redes convolucionales

Tabla 2: Descripción de las funciones de activación más utilizadas

2.3.5.3. Funciones de optimización.

Las funciones de optimización se utilizan para acercarnos a la mejor solución posible durante las diferentes etapas de entrenamiento de una red.

Estas son algunas de las funciones de optimización más usadas.

- Sgd (Stochastic Gradient Descent – Descenso del gradiente estocástico) (Instituto Tecnologico Autónomo de México, 2017): El descenso del gradiente es un algoritmo que estima dónde una función genera sus valores más bajos. En el caso de que el modelo de aprendizaje automático sea de gran escala este cálculo puede ser muy costoso. Debido a esto surge el descenso del gradiente estocástico que usa una constante y por consiguiente el número de gradientes a calcular es fijo.
- Momentum: es una variación de la función anterior. “Momentum” es un valor que acelera el descenso del gradiente si el signo del gradiente es el mismo durante diferentes épocas.
- AdaGrad (Velasco, 2020): introduce una variación en el concepto de tasa de entrenamiento. Esta función escala y adapta este valor para cada peso respecto del gradiente acumulado en cada iteración.
- Adadelta (Velasco, 2020): es una variación de AdaGrad en la que se restringe el cálculo de la tasa de entrenamiento de cada peso a una ventana de tamaño fijo de los últimos n gradientes en vez de hacerlo con el gradiente acumulado de cada iteración.
- RMSProp (Velasco, 2020) (Root Mean Square Propagation – Propagación de raíz cuadrática media): este algoritmo mantiene un factor de entrenamiento diferente para cada dimensión, pero el escalado del factor del entrenamiento se realiza dividiéndolo por la media del declive exponencial del cuadrado de los gradientes.
- Adam (Velasco, 2020) (Adaptive moment estimation – Estimación adaptativa del mo): Es una combinación de AdaGrad y RMSProp. Se mantiene un factor de entrenamiento por parámetro y se calcula

RMSProp, además cada factor de entrenamiento se ve afectado por el "momentum" del gradiente.

En la Figura 14 se muestra un gráfico animado en la que podemos ver unas comparativas de las diferentes funciones de optimización.

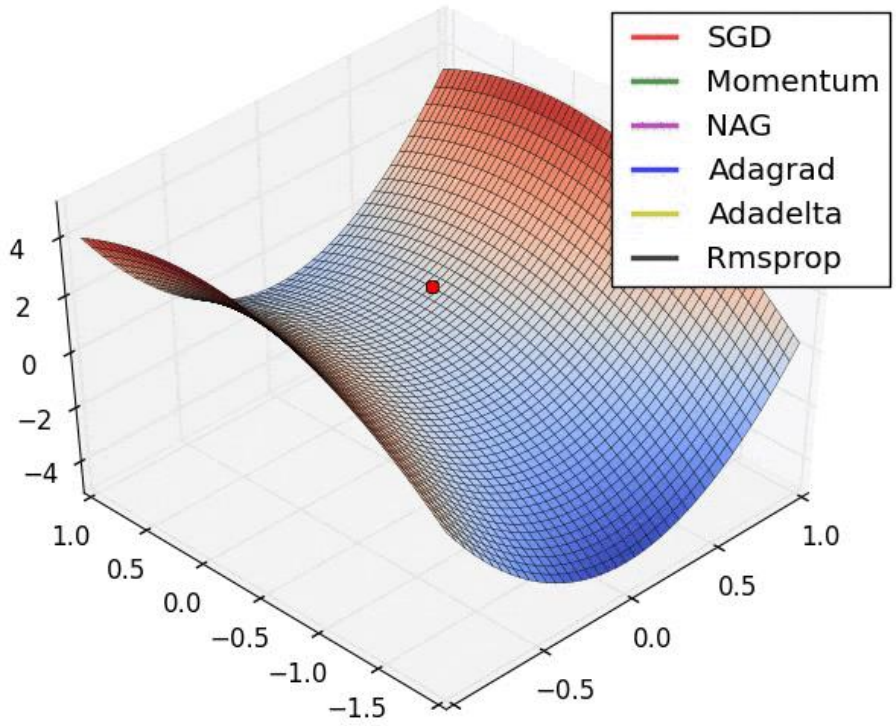

Figura 14: Comparativa de la convergencia de las diferentes funciones de optimización en funciones de coste de tipo valle profundo. *Fuente*

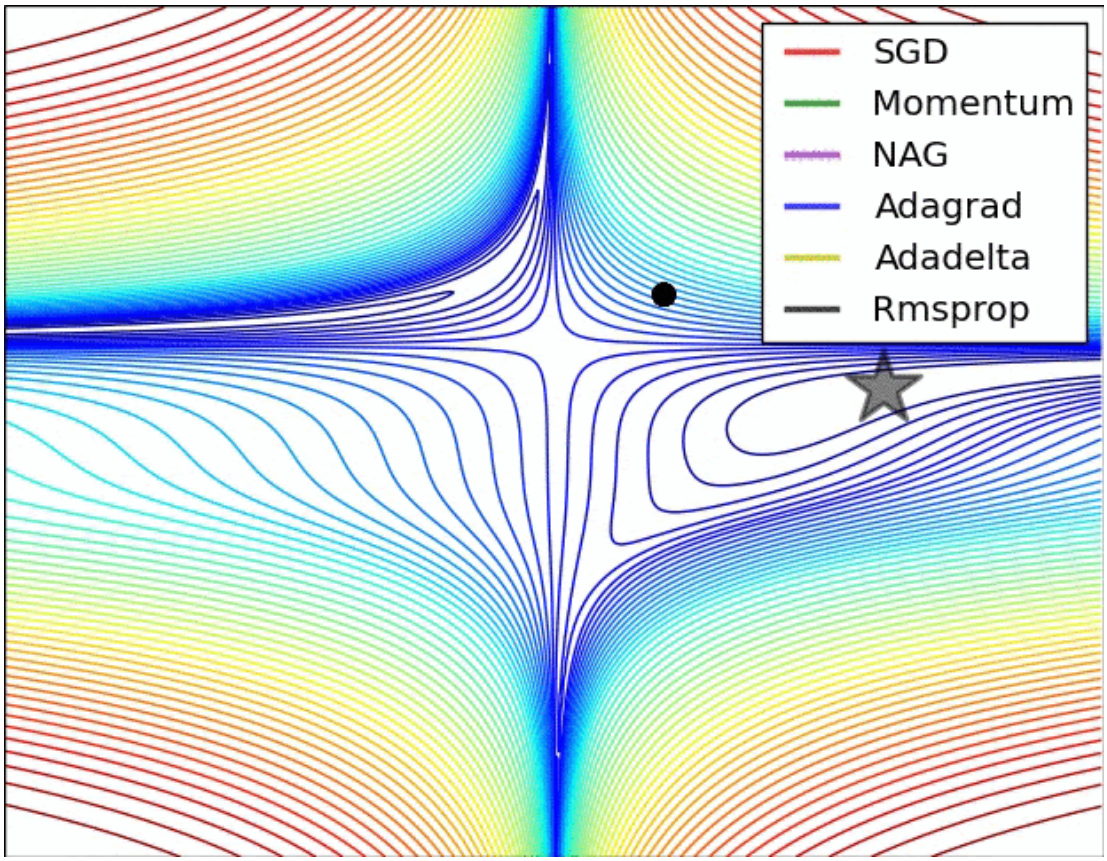

Figura 15: Comparativa de la convergencia de las diferentes funciones de optimización en funciones de coste tipo Beale. Fuente

Haciendo un análisis a simple vista de las funciones de optimización observamos en la Figura 15 que la bolita verde correspondiente a la función “Momentum” cuando ve que una dirección es la más correcta avanza a gran velocidad debido al valor de “momentum”. También observando la bolita roja correspondiente a “Sgd” es la más lenta y por consiguiente necesita un mayor número de etapas para llegar al objetivo. También se observa que la más rápida suele ser “Adadelta” que corresponde a la bolita amarilla gracias a que calcula la tasa de entrenamiento según los n últimos gradientes.

2.3.5.4. Funciones de pérdida.

Una función de pérdida, es una función que evalúa la desviación entre los diferentes valores obtenidos de predicciones del modelo con el valor real que corresponde. El objetivo es que el valor “loss” sea muy bajo lo que implica que la red neuronal es muy eficiente.

En tensorflow.js hay diferentes funciones de pérdida y la aplicación de cada una de ellas depende del tipo de problema que queramos afrontar.

- AbsoluteDifference (Acervolima, s.f.) (Diferencia absoluta): calcula en valor absoluto la distancia al objetivo. Esta función es utilizada en problemas de regresión.
- ComputeWeightedLoss (Acervolima, s.f.) (Media ponderada): calcula la pérdida ponderada entre dos tensores dados.
- CosineDistance (Acervolima, s.f.) (Distancia del coseno): calcula la pérdida aplicando el coseno de la distancia entre dos tensores.
- HingeLoss (Rennie & Srebro, 2005) (Pérdida de bisagra): calcula la pérdida de bisagra entre dos tensores. Se aplica en problemas de clasificación.
- HuberLoss (Pérdida Huber): es usada en problemas de regresión, esta calcula la pérdida provocada por el procedimiento de estimación.
- LogLoss (Shen, 2005) (Pérdida logarítmica): LogLoss o Logistic Loss es una función convexa que crece linealmente para números negativos y la hace poco sensible a valores atípicos. Calcula la pérdida logarítmica entre dos tensores.
- MeanSquaredError (Acervolima, s.f.)(Error cuadrático medio): calcula de forma geométrica la distancia al cuadrado al objetivo, esta función es utilizada en problemas de regresión.
- CategoricalCrossEntropy (Acervolima, s.f.): mide la distancia entre distribuciones de probabilidad. Es aplicada en redes cuya capa de salida es una probabilidad.
- SigmoidCrossEntropy (Acervolima, s.f.): se trata de una variante de la función “CategoricalCrossEntropy”, en este caso calcula la pérdida en un valor entre 0 y 1.
- SoftmaxCrossEntropy (Bendersky, 2016): se aplica una combinación de las funciones softmax y CrossEntropy calculando la pérdida de la red.

2.3.5.5. Métricas.

- BinaryAccuracy (Acervolima, s.f.) (Precisión binaria): calcula la frecuencia con la que las predicciones coinciden con las etiquetas binarias.
- BinaryCrossentropy (Saxena, 2021) (Entropía cruzada binaria): es el promedio negativo del logaritmo de las probabilidades predichas que se han corregido.
- CategoricalAccuracy (Exactitud categórica): calcula la frecuencia con la que las predicciones coinciden con las etiquetas one-hot.
- CategoricalCrossentropy (Entropía cruzada categórica): calcula la métrica de entropía cruzada entre las etiquetas y las predicciones.
- CosineProximity (Proximidad del coseno): calcula entre las etiquetas y las predicciones el coseno de la proximidad. Normalmente se obtienen valores negativos.
- MeanAbsoluteError (Error absoluto medio): en estadística, es una medida de la diferencia entre dos variables continuas, aplicada a los modelos de aprendizaje automático estas variables son las etiquetas y las predicciones.
- MeanAbsolutePercentageError (Porcentaje de error absoluto medio): de igual forma que “MeanAbsoluteError” calcula la diferencia entre las etiquetas y las predicciones, pero en este caso con un porcentaje.
- MeanSquaredError (Error cuadrático medio): determina el error cuadrático medio entre las etiquetas y las predicciones.
- Precisión
- Recall
- SparseCategoricalAccuracy

2.4. Restricciones.

El **TFG** se define como una asignatura de 12 créditos. Estos 12 créditos se resumen en 300 horas de proyecto teniendo en cuenta todas las etapas del ciclo de vida del proyecto a excepción del mantenimiento. Por consiguiente, la principal restricción de este proyecto es las limitadas horas de trabajo.

Otra restricción del proyecto está relacionada con el entrenamiento de modelos en el proyecto. Como hemos mencionado anteriormente los modelos se entrenan desde la web recayendo el coste computacional en la máquina del usuario final, debido a esto no se pueden crear y entrenar modelos muy complejos.

2.5. Metodología de desarrollo software.

Hay multitud de metodologías en el desarrollo del software, a continuación, se muestra una comparativa de las metodologías más utilizadas.

2.5.1. Comparativa de metodologías.

La metodología es un apartado fundamental a la hora de desarrollar un proyecto, por ello y debido a que existen diferentes metodologías es necesario realizar un estudio de la más adecuado teniendo en cuenta el proyecto que se va a realizar.

Hay una diferenciación entre las metodologías tradicionales y las metodologías ágiles.

A continuación, voy a mostrar algunos de las metodologías tradicionales más populares:

Modelo	Ventajas	Desventajas
En cascada	Modelo lineal Los costes y la carga de trabajo se pueden estimar al comenzar el proyecto	El proceso de definición y desarrollo de software es no-lineal No hay demos hasta etapas avanzadas del proyecto Incertidumbre al comienzo
Incremental	El cliente puede usar el software en versiones tempranas	Dificultad para determinar los requisitos

	Uso de primeros incrementos para obtener información Objetivos prioritarios son más probados	Cuando se comienza un incremento no se pueden modificar los requisitos
Desarrollo Rápido de Aplicaciones (DRA)	Mejora del modelo en cascada Componentes reutilizables Recomendable para aplicaciones con un uso intensivo de datos	En sistemas grandes es complejo de organizar Dificultad al salir de los estándares Problemas de rendimiento al cargar funcionalidad no necesaria
En espiral (IONOS, 2020)	Control sobre costes y recursos Modelo flexible	No apropiado para proyectos pequeños Gran esfuerzo de gestión

Tabla 3: Comparativa entre modelos tradicionales

Ahora en la Tabla 4 analizamos las diferencias entre las metodologías ágiles más comunes:

Metodología	Ventajas	Desventajas
Scrum (Drew, 2019)	Resultados anticipados Flexibilidad, adaptación de los contextos Gestión sistemática de riesgos	Para equipos reducidos Requiere buena definición de las tareas y plazos Requiere alta cualificación

Kanban (Mas, 2019)	Buena organización del flujo de trabajo Evita la acumulación de trabajo Plazos de entrega continuo	Malo para aumentos de demanda Para proyectos grande tiene un alto coste de almacenamiento
Desarrollo dirigido por test (TDD) (Grifol)	Requiere una organización meticulosa Unifica el proceso de depuración con el desarrollo	Recomendada para proyectos con lenguajes compilados Problemas en el desarrollo con varias personas

Tabla 4: Comparativa de las metodologías ágiles más usadas

Una de las principales diferencias entra las metodologías ágiles y las metodologías tradicionales es que las metodologías ágiles están más enfocadas a equipos de trabajo y debido a esto no las vamos a tener en cuenta para el desarrollo de este proyecto.

En la Tabla 3 se muestran múltiples modelos de desarrollo, de todos ellos el más conocido y antiguo es el modelo en cascada, pero es el menos flexible. Por ello y qué, además, se va a realizar un modelo relativamente pequeño se ha decidido realizar el proyecto siguiendo la metodología incremental.

2.5.2. Desarrollo incremental.

El desarrollo incremental se trata de un proceso de desarrollo de software que se creó para suprimir las debilidades del modelo de desarrollo tradicional en cascada.

El proyecto se divide en diferentes planificaciones, cada una de ellas se llama iteración. En cada iteración se repiten un conjunto de pasos que aportan un resultado más completo sobre el producto final, de forma que el usuario final va recibiendo poco a poco el producto de forma creciente.

Para lograr esto, en cada iteración debe de haber unos requerimientos únicos acompañados de pruebas y documentación para su correcta implementación. En cada

iteración los componentes evolucionan agregando más opciones de requisitos y logrando así una mejora más completa.

Debido a que en este Trabajo de Fin de Grado se va a entregar un producto software, se ha considerado que lo más adecuado es para el desarrollo es utilizar un modelo de desarrollo incremental, de esta forma se pueden realizar mejoras de funcionalidad poco a poco y los requisitos más básicos (requisitos iniciales) están muy depurados.

2.5.3. Ciclo de vida del desarrollo incremental.

Los pasos claves de este proceso son comenzar con una implementación simple que sirva de base para el resto del desarrollo e iterativamente ir complicando la implementación y finalizando todas las funcionalidades requeridas por el cliente.

El modelo incremental se divide en dos etapas, la etapa de iniciación y la etapa de iteración. En la etapa de iniciación se crea una versión del sistema. La meta de esta etapa es crear un producto con el que el cliente pueda interactuar y probar el software. De esta forma desde el primer momento se está retroalimentando el proceso. En la etapa de iteración se realiza un rediseño e implementación en base al feedback del incremento probado por el cliente. La meta de esta etapa es ser simple, directa y modular para soportar el rediseño (Figura 16).

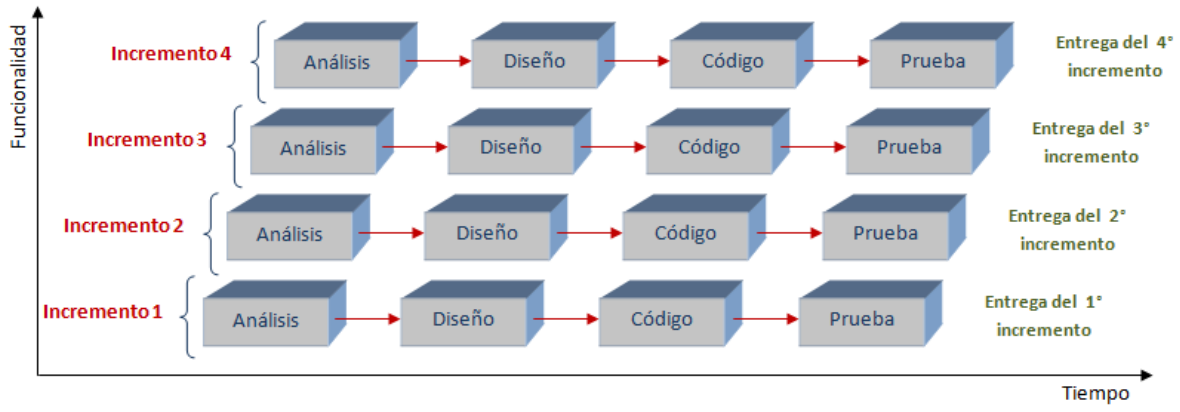

Figura 16: Metodología de desarrollo incremental. Fuente

2.6. Normas y referencias.

El proyecto desarrollado hace uso de imágenes de forma local, de forma que no guarda nada en la parte del servidor, por lo tanto, no se viola la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) (Estado, 2018).

En este proyecto se han utilizado las normas UNE 157001:2014 “Criterios generales para la elaboración de proyectos” y UNE 157801:2007 “Criterios generales para la elaboración de proyectos de sistemas de información”. Estas normas se han utilizado como base o como referencia para la inclusión de algunos contenidos y definiciones sobres la elaboración de proyectos. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.

Debido a que el proyecto es un proyecto de ingeniería se debe aplicar la norma UNE 157001 “Criterios generales para la elaboración de proyectos”, pero aplicaremos la norma UNE 157801 “Criterios generales para la elaboración de proyectos de Sistemas de Información”.

3. ANÁLISIS Y PLANIFICACIÓN.

Un buen diseño software permite el buen entendimiento de los requisitos haciendo que el software tenga la funcionalidad necesaria, creando un sistema robusto y en el que sus piezas se pueden usar en otros sistemas. Siempre un buen diseño ayuda a conseguir un buen software, aunque un buen diseño no implica un buen software final.

El sistema se trata de una aplicación web la cual permite al usuario adentrarse en el campo de la inteligencia artificial, más concretamente en la rama del "Deep learning" o aprendizaje profundo (Goodfellow, Bengio, & Courville, 2016).

3.1. Planificación temporal.

Para completar los objetivos del presente proyecto, debemos de establecer una planificación temporal que determine la tarea que se va a realizar en cada período de la duración total del proyecto. En la Tabla 5 se recogen las diferentes tareas a realizar y la estimación de tiempo necesaria para realizarla, cabe destacar que estos tiempos se tratan de una estimación y por lo tanto los tiempos son orientativos.

Tarea a realizar	Duración estimada
Estudio y formación en redes de aprendizaje profundo	8 semanas
Estudio y formación JavaScript y React.js	8 semanas
Estudio y formación Tensorflow.js	4 semanas
Desarrollo del software	10 semanas
Elaboración de la documentación	5 semanas
Total	**35 semanas**

Tabla 5: Planificación temporal del proyecto

Debido a que el modelo de desarrollo elegido es el modelo incremental, la parte de desarrollo del software y la fase de pruebas del mismo no se separa, es decir, estas fases se irán intercalando.

Para visualizar de forma más sencilla la estimación de la tabla anterior, vamos a mostrar la misma estimación, pero con el diagrama de Grant (Tabla 6,Tabla 7,Tabla 8)

		Semana											
Tarea	**Duración**	1	2	3	4	5	6	7	8	9	10	11	12
Redes	8 sem.												
JS y React.js	8 sem.												
Tensorflow.js	4 sem.												
Desarrollo	10 sem.												
Documentación	5 sem.												

Tabla 6: Diagrama de Grant (I)

	Semana													
Tarea	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
Redes															
JS y React.js															
Tensorflow.js															
Desarrollo															
Documentación															

Tabla 7: Diagrama de Gran (II)

	Semana								
Tarea	28	29	30	31	32	33	34	35	
Redes									
JS y React.js									
Tensorflow.js									
Desarrollo									
Documentación									

Tabla 8: Diagrama de Grant (III)

3.2. Presupuesto.

Para estimar el coste de este proyecto se han de tener en cuenta el apartado del software utilizado como el hardware y los costes asociados al desarrollo del mismo.

3.2.1. Hardware.

El hardware requerido para el desarrollo y prueba del proyecto conlleva unos costes asociados. En este caso el desarrollo y prueba del software se ha realizado desde los mismos equipos: un equipo portátil y un equipo de sobremesa (Tabla 9,Tabla 10).

Ordenador sobremesa	
Componente	**Precio**
AMD Ryzen 2600	150,00 €
Sapphire nitro AMD RX580 8GB	200,00 €
16 GB Memoria RAM (2x16GB)	80,00 €
1 TB disco duro Seagate	39,65 €
Caja, refrigeración, periféricos, etcétera	200,00 €
Monitor SAMSUNG	150,00 €
Total	**819,65 €**

Tabla 9: Estimación de costes del hardware de desarrollo (I)

Ordenador portátil	
Componente	Precio
ASUS A-555LJ 2016	800,00 €
Ratón Bluetooth	10,00 €
Total	**810,00 €**

Tabla 10: Estimación de costes del hardware de desarrollo (II)

Total, hardware de desarrollo	
Equipo	**Precio**
Ordenador sobremesa	819,65 €
Ordenador portátil	810,00 €
Total	**1629,65 €**

Tabla 11: Estimación de costes del hardware de desarrollo (III)

Si el coste total de los equipos es de 1629,65 € (Tabla 11), y supóngase que el gasto en los mismos es una cantidad que se amortiza en 5 años, necesitarían pagarse $\frac{1629{,}65\ €}{5}$ = 325,93 €/año para amortizar los equipos. Dado que el proyecto tiene una duración total de 8,75 meses (35 semanas), en ese período se amortizaría una cantidad de 325,93 €/año * 12 meses/año * 8,75 meses = 237,65 €, lo que supone una fracción del total de $\frac{237{,}65\ €}{1629{,}65\ €} \times 100 = 14{,}58\%$ amortizado de los equipos.

3.2.2. Software y datos.

Para el desarrollo, test y realización de la memoria se han utilizado tanto softwares privativos como públicos.

Software/Recurso	Precio
Licencias Windows 10 incluidas con hardware	0,00 €
Visual Paradigm (Licencia con cuenta UJA)	0,00 €
GitHub	0,00 €
JavaScript	0,00 €
React.js	0,00 €
Visual Studio Code	0,00 €
Total	0,00 €

Tabla 12: Estimación de costes del software

No se calcula la amortización para el coste del software porque el coste es nulo (Tabla 12).

3.2.3. *Recursos humanos.*

El proceso de desarrollo durará 35 semanas de las cuales habrá 175 días laborales. Teniendo en cuenta que el salario bruto de un programador es 24.640,37 € (Boletín oficial del estado, 2018) se ha realizado una estimación del coste en recursos humanos del desarrollo del proyecto véase Tabla 13.

Concepto	Duración	Precio
Programador (media jornada)	1 mes	1.026,68 €
Total	**8,75 meses**	**8.983,46 €**

Tabla 13: Estimación de costes de recursos humanos

3.2.4. *Costes indirectos.*

En los costes indirectos se incluyen todos los gastos derivados del desarrollo como pueden ser la vivienda, luz, internet… (Tabla 14)

Concepto	Tarifa	Precio
Consumo eléctrico	40,00€/mes	350,00 €
Acceso a internet	30,00€/mes	262,50 €
Total		**612,50 €**

Tabla 14: Estimación de costes indirectos

3.2.5. *Estimación total.*

Concepto	Precio
Hardware (p. amortizado en los 8,75 meses de duración)	237,65 €
Software	0,00 €
Recursos humanos	8.983,46 €
Costes indirectos	612,50 €
Total	**9.833,61 €**

Tabla 15: Estimación total de costes

Teniendo en cuenta que la duración del proyecto es de 8,75 meses, el gasto de cada mes es de: 1.123,84 € y el gasto total es 9.833,61 € (Tabla 15).

3.3. Captura de requisitos.

Para fijar los requisitos funcionales del sistema, los requisitos se han plasmado mediante un diagrama de casos de uso (Figura 17). El diagrama de casos de uso permite reflejar el comportamiento del sistema desde el punto de vista del cliente.

En la fase inicial del desarrollo estos casos de uso deben ser utilizados para construir la base del proyecto a partir de la cual y a través de los diferentes incrementos se irá mejorando el diseño y comportamiento de nuestro sistema.

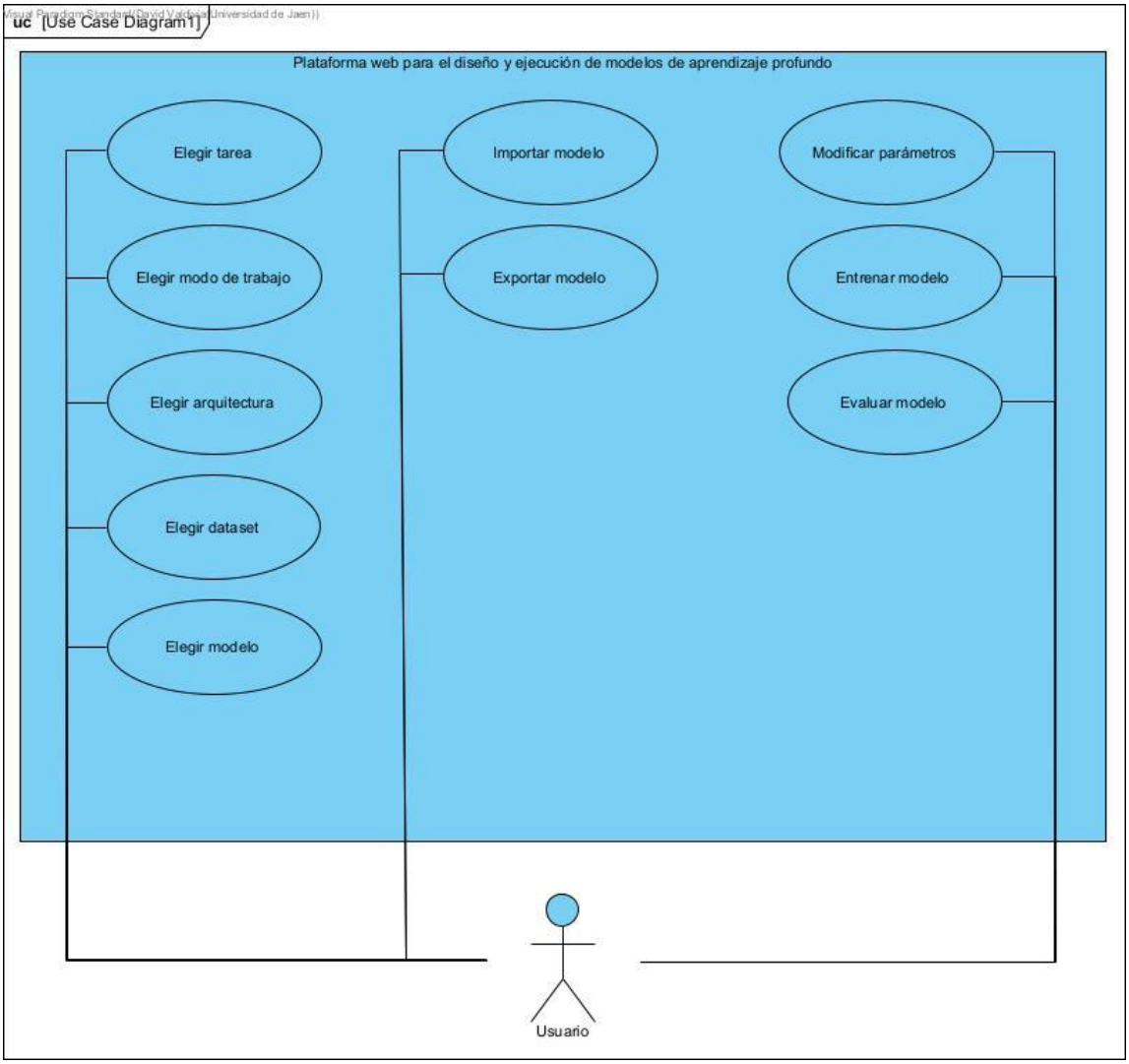

Figura 17: Diagrama de casos de uso del sistema

3.4. Requisitos funcionales.

Primero vemos los diferentes casos de uso que tiene nuestra aplicación (Tabla 16)

Código	Caso de uso
CU-01	Elegir tarea
CU-02	Elegir modo de trabajo
CU-03	Elegir dataset
CU-04	Elegir arquitectura
CU-05	Elegir modelo
CU-06	Importar modelo
CU-07	Exportar modelo
CU-08	Modificar parámetros
CU-09	Entrenar modelo
CU- 10	Evaluar modelo

Tabla 16: Requisitos funcionales

En el primer Caso de uso (Tabla 17), se define la interacción en la que el usuario selecciona el tipo de tarea dentro de la aplicación. Esta tarea nos sirve para seleccionar entre los diferentes tipos de redes de aprendizaje automático.

CU-01	Elegir tarea
Actores	Usuario, sistema
Precondiciones	- Cargar página web
Postcondiciones	- Ninguna
Escenario principal	1. El usuario selecciona uno de los botones del menú inicial 2. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	- Ninguno

Tabla 17: CU-01 Elegir tarea

El CU-02 (Tabla 18) permite una vez que hemos elegido al tarea o finalidad seleccionar si queremos usar un modelo que ya ha sido pre-entrenado, simplemente para ponerlo en práctica o si queremos empezar a crear un modelo eligiendo su arquitectura.

CU-02	Elegir modo de trabajo
Actores	Usuario, sistema
Precondiciones	- Haber seleccionado una tarea
Postcondiciones	- El sistema muestra al usuario dos opciones: o Modelo Pre-entrenado o Crear/Editar arquitectura
Escenario principal	1. El usuario selecciona el modo de trabajo 2. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	- Ninguno

Tabla 18: CU-02 Elegir modo de trabajo

Si en CU-02 hemos elegido Crear/Editar arquitectura, ahora, en el CU-03 (Tabla 19) tendremos la opción de elegir el dataset que se va a emplear para crear el modelo. El usuario puede elegir entre un dataset proporcionado por la aplicación o el usuario puede subir uno propio.

CU-03	Elegir dataset
Actores	Usuario, sistema
Precondiciones	- Haber seleccionado una tarea - Haber elegido “Crear/Editar arquitectura” en el CU-02
Postcondiciones	- El sistema muestra al usuario el listado de datasets
Escenario principal	1. El usuario selecciona el dataset de trabajo

	2. Fin del escenario con éxito
Escenarios alternativos	1.a No hay opciones disponibles 1. El sistema no muestra opciones en el desplegable 2. Volver al paso 1 de escenario principal
Escenarios de excepción	- Ninguno

Tabla 19: CU-03 Elegir dataset

Una vez hemos elegido el dataSet de trabajo, en CU-04 (Tabla 20) tenemos la opción de seleccionar una arquitectura a partir de un fichero de nuestra máquina o de usar una precargada por la aplicación.

CU-04	**Elegir arquitectura**
Actores	Usuario, sistema
Precondiciones	- Haber seleccionado una tarea - Haber seleccionado un dataset
Postcondiciones	- Tras finalizar pulsar el botón “Continuar”
Escenario principal	1. El usuario pulsa el botón para cargar arquitectura 2. Navega entre su sistema de archivos para cargar un archivo de tipo “.json” 3. Selecciona el fichero deseado y acepta la subida 4. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	2.a No selecciona ningún archivo y cancela la carga del fichero 1. Se vuelve al estado 1 del escenario principal 3.a Fichero no compatible 1. Se vuelve al estado 1 del escenario principal

Tabla 20: CU-04 Elegir arquitectura

Si en CU-02 hemos elegido Modelo Pre-entrenado, ahora, en el CU-05 (Tabla 21) tendremos la opción de elegir un modelo proporcionado por la aplicación para

hacer pruebas con él. El usuario puede elegir una de las opciones del desplegable, siendo una de ellas la carga de un modelo personalizado.

CU-05	Elegir modelo
Actores	Usuario, sistema
Precondiciones	- Haber seleccionado una tarea - Haber elegido “Modelo Pre-entrenado” en el CU-02
Postcondiciones	- Tras finalizar pulsar el botón “Continuar”
Escenario principal	1. El usuario selecciona uno de las opciones del desplegable 2. Fin del escenario con éxito
Escenarios alternativos	1.a El usuario selecciona cargar su propio modelo 1. Pulsa el botón continuar 2. En la siguiente vista puede cargar su modelo subiendo el fichero “.bin” y “.json” necesarios
Escenarios de excepción	- Ninguno

Tabla 21: CU-05 Elegir modelo

La aplicación permite la importación de un modelo solicitando dos ficheros en formato “.bin” y “.json”. Para acceder a esta importación el usuario debe de elegir la opción de subir modelo propio en CU-05 y realizar CU-06 (Tabla 22) en la siguiente vista.

CU-06	Importar
Actores	Usuario, sistema
Precondiciones	- Haber seleccionado cargar modelo en CU-05
Postcondiciones	- La aplicación carga el modelo a partir de los ficheros

Escenario principal	1. El usuario pulsa el botón para cargar arquitectura 2. Navega entre su sistema de archivos para cargar un archivo de tipo “.json” 3. Selecciona el fichero deseado y acepta la subida 4. El usuario pulsa el botón para cargar datos 5. Navega entre su sistema de archivos para cargar un archivo de tipo “.bin” 6. Selecciona el fichero deseado y acepta la subida 7. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	2.a No selecciona ningún archivo y cancela la carga del fichero 1. Se vuelve al estado 1 del escenario principal 3.a Fichero no compatible 1. Se vuelve al estado 1 del escenario principal 5.a No selecciona ningún archivo y cancela la carga del fichero. 1. Se vuelve al estado 1 del procesamiento 6.a Fichero no compatible 1. Se vuelve al estado 1 del escenario principal

Tabla 22: CU-06 Importar

En el CU-07 (Tabla 23), se le permite al usuario exportar el modelo que ha creado con la aplicación. Esta opción es visible a el usuario cuando finaliza el entrenamiento de una red.

CU-07	Exportar
Actores	Usuario, sistema
Precondiciones	- Entrenar una red con la aplicación
Postcondiciones	- Ninguna
Escenario principal	1. El usuario pulsa el botón “Exportar modelo” 2. Aparece una ventana del explorador de archivos para guardar un archivo con extensión “.json” 3. El usuario elige un lugar de destino y pulsa el botón guardar 4. Aparece una ventana del explorador de archivos para guardar un archivo con extensión “.bin” 5. El usuario elige un lugar de destino y pulsa el botón guardar 6. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	3.a Error al guardar 1. Se continúa en el escenario principal 4 5.a Error al guardar 1. Se continua en el escenario principal 6

Tabla 23: CU-07 Exportar

Tras el caso CU-04 nos aparece una nueva vista. En esta vista se va a realizar todo lo referente a crear el modelo, modificarlo y entrenarlo. El caso CU-08 (Tabla 24) trata de como el usuario puede modificar los parámetros de la arquitectura.

CU-08	Modificar parámetros
Actores	Usuario, sistema
Precondiciones	- Haber elegido una arquitectura

Postcondiciones	- Ninguna
Escenario principal	1. El usuario modifica algún dato del formulario 2. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	- Ninguno

Tabla 24: CU-08 Modificar parámetros

El CU-09 (Tabla 25) permite al usuario entrenar un modelo en base a los parámetros del formulario que pueden ser modificados con el caso CU-08. Tras entrenarlo el modelo está listo

CU-09	**Entrenar modelo**
Actores	Usuario, sistema
Precondiciones	- Haber elegido una arquitectura
Postcondiciones	- Tras entrenar modelo aparece el botón para exportarlo (CU-07)
Escenario principal	1. Pulsar el botón “Entrenar modelo” 2. Fin del escenario con éxito
Escenarios alternativos	- Ninguno
Escenarios de excepción	1.a Error al entrenar le modelo 1. Revisar parámetros de entrenamiento (CU-08)

Tabla 25: CU-09 Entrenar modelo

En CU-10 (Tabla 26), se define la interacción en la que el usuario evalúa el modelo creado o seleccionado mostrando el sistema una alerta con la predicción que ha realizado el modelo con respecto los datos de entrada.

CU-10	Evaluar modelo
Actores	Usuario, sistema
Precondiciones	- Haber entrenado un modelo (CU-09) o haber elegido “Modelo pre-entrenado” en CU-02
Postcondiciones	- Ninguna
Escenario principal	1. El usuario rellena el formulario de evaluación 2. El usuario pulsa el botón evaluar modelo 3. El sistema muestra una alerta con el resultado de la predicción
Escenarios alternativos	1.a La entrada puede ser seleccionar una imagen 3.a Si la entrada es una imagen o vídeo 1. La predicción del modelo se muestra sobre la imagen o vídeo de la entrada
Escenarios de excepción	- Ninguno

Tabla 26: CU-10 Evaluar modelo

3.5. Requisitos no funcionales.

Son los requisitos que describen características del funcionamiento del sistema y no las funciones que realiza. A continuación, se muestran los requisitos no funcionales:

1. El nuevo sistema debe desarrollarse aplicando patrones y recomendaciones de programación.
2. La aplicación debe tener interfaces intuitivas, de fácil uso.
3. El sistema deberá tener una interfaz responsive.
4. El sistema debe contar con un manual de usuario.
5. El sistema debe ser tolerante ante fallos.

4. DISEÑO

Ahora, gracias a los casos de uso, podremos hacer los diagramas de secuencia y el diagrama de clases del proyecto.

4.1. Diagramas de secuencia.

A partir de los casos de uso y el modelo de dominio podemos realizar los diagramas de secuencia, los cuales nos permiten diseñar la interacción que va a tener el usuario con el software.

Hacer una gran cantidad de diagramas de secuencia puede generar aspectos negativos como un gran esfuerzo en la etapa de diseño o hacer un sistema más complejo, pero todo el tiempo empleado en el diseño será tiempo ahorrado en problemas durante el desarrollo.

En los diagramas se diferencia la Vista Usuario del resto de la aplicación, y se define como una interfaz con la que el usuario interactúa.

En el primer diagrama de secuencia (Figura 18) se muestra como el usuario inicialmente debe de elegir una tarea de las disponibles siempre y cuando el sistema muestra la página correctamente. Es el primer caso de uso que se va a encontrar el usuario cuando acceda a la web.

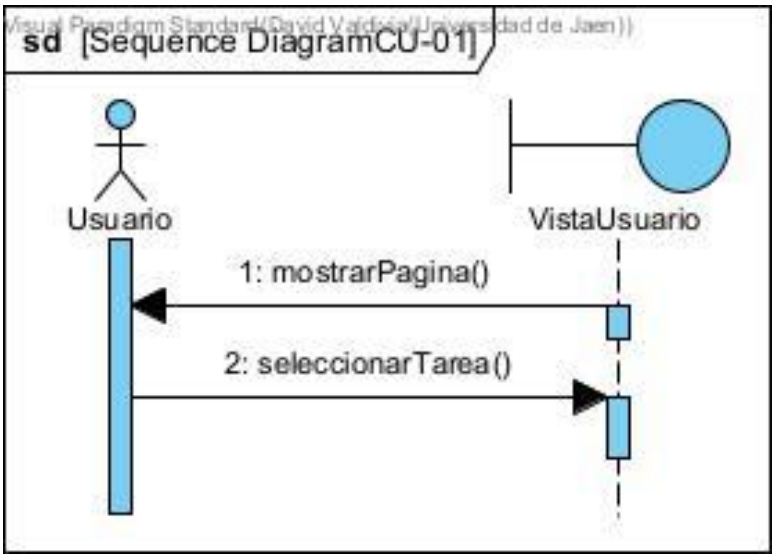

Figura 18: Diagrama de secuencia del caso de uso CU-01

Tras el primer caso de uso, donde el usuario elige el tipo de modelo que se va a desarrollar, el usuario debe de elegir entre dos categorías (Figura 19). La primera de ellas es “Modelo pre-entrenado”, esta categoría nos permite importar un modelo pre-entrenado para que el usuario pueda evaluarlo directamente. La segunda categoría corresponde a “crear/editar arquitectura”. Aquí el sujeto puede crear una arquitectura modificando los parámetros de un ejemplo. También elige el set de datos que se va a usar para entrenar la arquitectura.

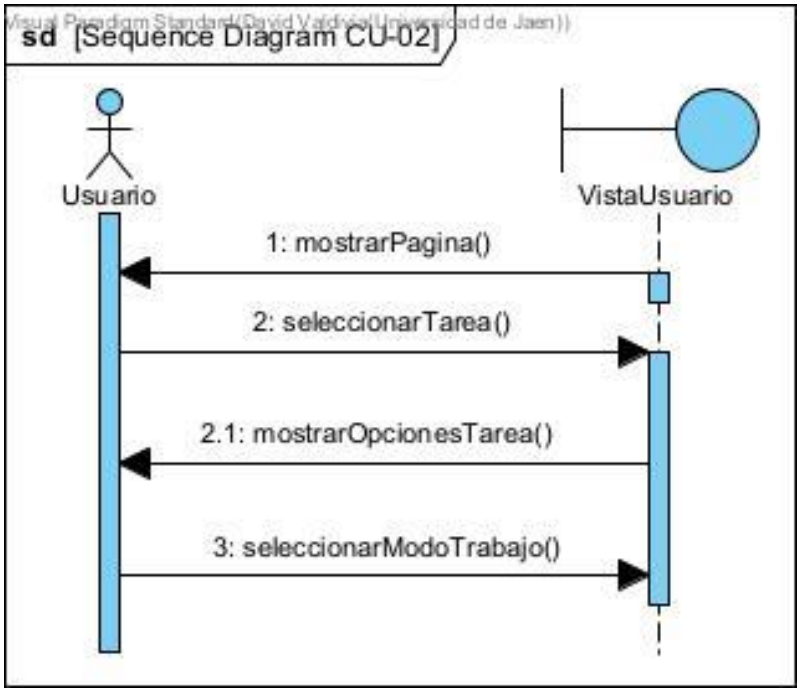

Figura 19: Diagrama de secuencia del caso de uso CU-02

En la Figura 20 vemos el diagrama de secuencia del caso de uso 3. Este diagrama resume la interacción entre el usuario y la interfaz a la hora de elegir un dataset. Para llegar a este punto se ha debido de pasar por el caso de uso 2 y elegir la opción “Crear/editar arquitectura”.

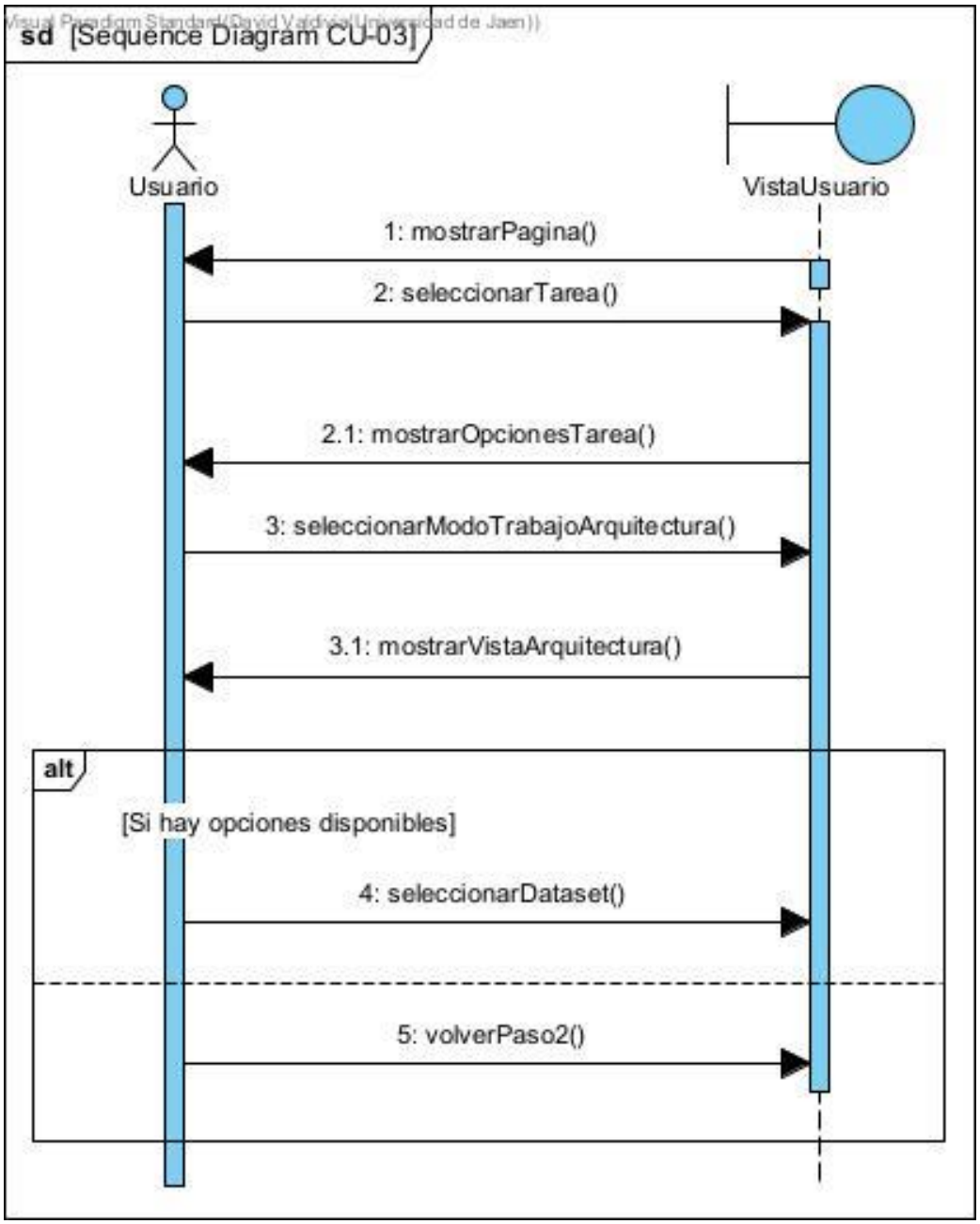

Figura 20: Diagrama de secuencia del caso de uso CU-03

El caso de uso 4 es “Elegir arquitectura”. En el siguiente diagrama de secuencia (Figura 21) se muestra la secuencia de acciones que se requieren para elegir una arquitectura del usuario. Como en la Figura 19 partimos del caso de uso 2 eligiendo “crear/editar arquitectura”, tras ello se le da la opción al usuario de subir su propia arquitectura pulsando un botón y seleccionando un archivo de su equipo. El sistema se encarga de validar el archivo y de que no haya ningún problema a la hora de importarlo. Si el fichero no tiene el formato correcto o no tiene una estructura correcta

el sistema emitirá una alerta comunicando al usuario que la importación no se ha realizado con éxito.

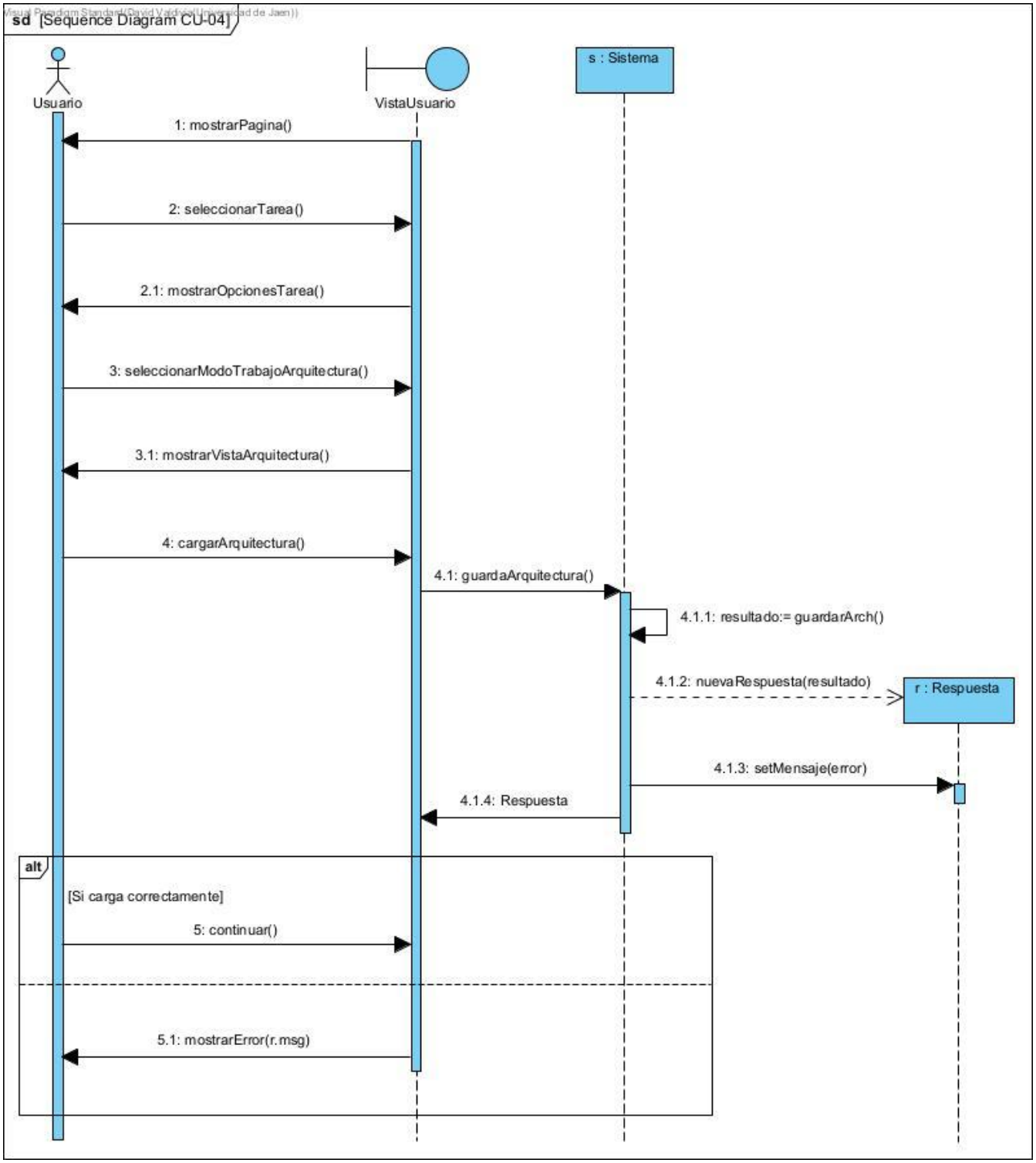

Figura 21: Diagrama de secuencia del caso de uso CU-04

La Figura 22 define como el usuario selecciona un modelo para más tarde poder evaluarlo. Partimos del caso de uso 2 donde el usuario debe de elegir la tarea y la opción “Modelo pre-entrenado”. Ahora se refresca la interfaz y muestra al usuario un desplegable donde elige entre los diferentes modelos, incluso se le da la opción de que el usuario suba su propio modelo (caso de uso 6). Tras seleccionar una de las opciones del desplegable el usuario debe de pulsar el botón de continuar.

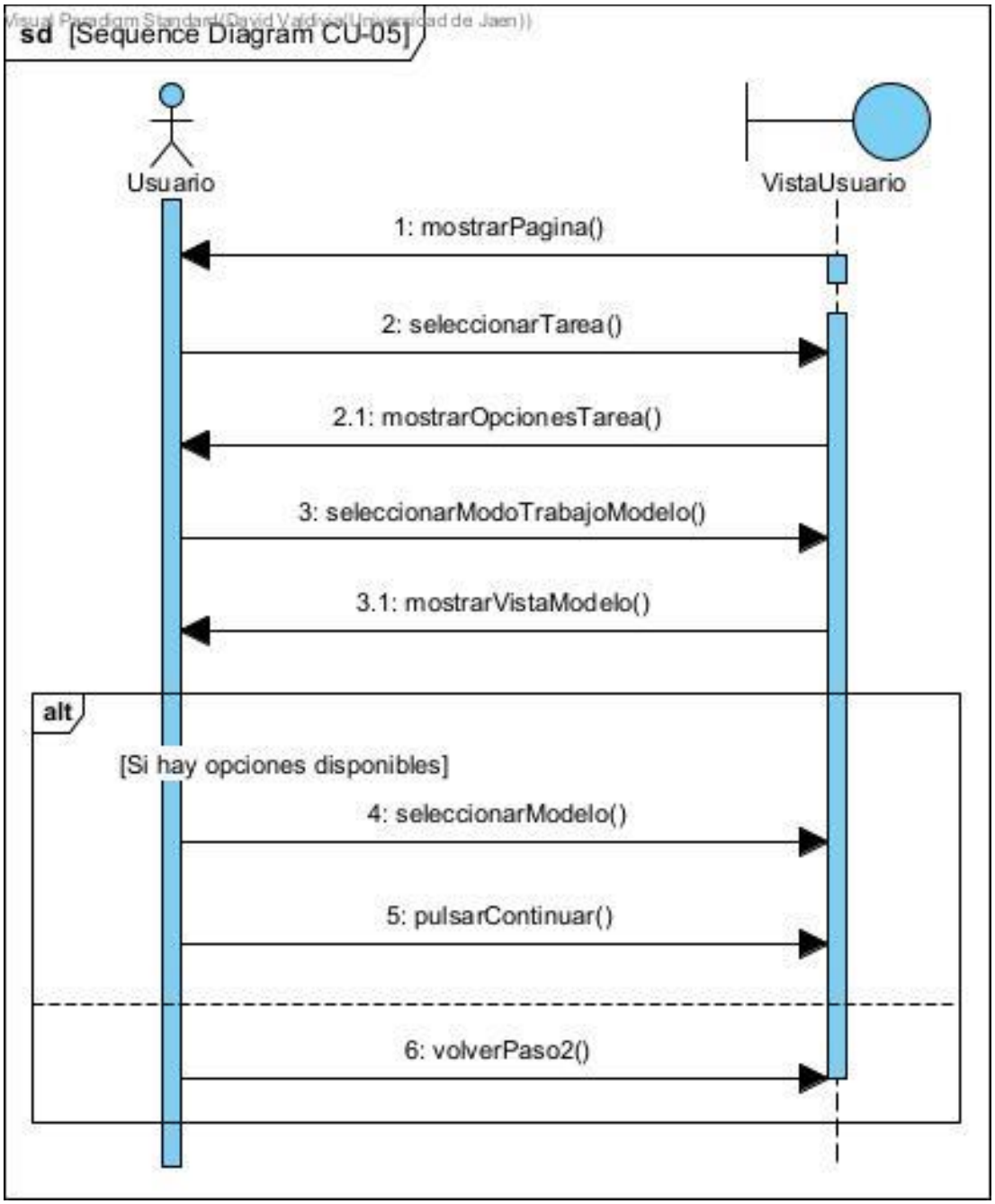

Figura 22: Diagrama de secuencia del caso de uso CU-05

El diagrama de secuencia de la Figura 23 se corresponde con la importación de modelos. Este es uno de los diagramas de secuencia del que se compone la aplicación ya que se necesita que el usuario cargue varios ficheros y que estos tengan un formato específico con una estructura específica. Partimos del caso de uso 5 donde el usuario tiene que seleccionar un modelo, como se ha comentado una de las opciones de ese desplegable es “SUBIR MODELO PROPIO”. El usuario debe de seleccionar esta opción y pulsar el botón continuar.

Cuando la web actualiza la interfaz en la parte superior de la página vemos diferentes botones para que el usuario pueda subir los diferentes ficheros necesarios para importar un modelo. El primer fichero se trata de un “.**JSON**”, este fichero proporciona toda la información acerca de la estructura del modelo y los pesos. Es muy importante que tengan una estructura válida, de lo contrario la importación no se podrá realizar con éxito. El segundo fichero es un archivo “**.BIN**”, contiene la información de los datos con los que se ha entrenado el modelo. Para crear un modelo son necesario obligatoriamente estos dos archivos. Como se ha mencionado estos archivos necesitan una estructura específica, desde tensorflow.js se pueden utilizar varias funciones tanto para la importación como la exportación de modelos y arquitecturas, en este caso la función para la importación necesita de estos dos ficheros.

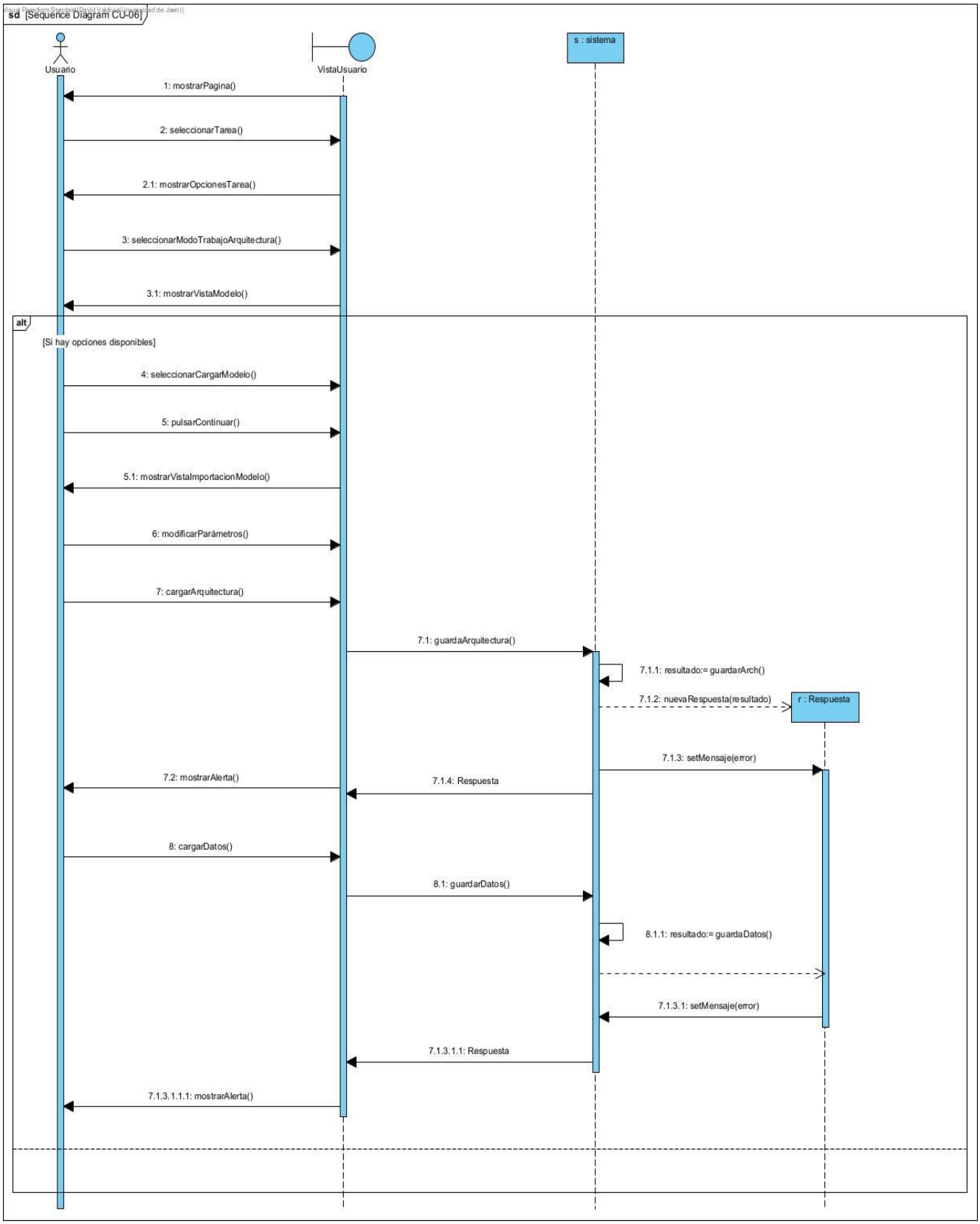

Figura 23: Diagrama de secuencia del caso de uso CU-06

Al igual que el usuario puede importar una arquitectura o un modelo, la aplicación permite al usuario exportar el modelo que el usuario ha creado. Para explicar la interacción que debe usar el usuario tenemos la Figura 24. El usuario tras elegir una tarea y el modo de trabajo “crear/editar arquitectura” se le muestra una interfaz donde puede modificar la arquitectura y entrenar el modelo. Al acceder a esta sección de la web se le proporciona al usuario una arquitectura y un formulario relleno, de forma que el usuario si no tiene conocimientos pueda crear su primera red de aprendizaje automático. Tanto la arquitectura como el formulario se pueden modificar como se mostrará en el caso de uso 8.

Antes de poder exportar la red, esta necesita ser entrenada y para ello el usuario debe de pulsar el botón “Crear y entrenar modelo”. Si el modelo se ha entrenado con éxito la web mostrará un nuevo botón “Exportar modelo”. Para exportar el modelo el usuario pulsará este botón y automáticamente se abrirá el explorador de archivos de su equipo pidiendo un lugar para almacenar los diferentes ficheros.

Como se ha comentado en el apartado anterior, se usa una función de tensorflow.js para la exportación de los modelos, esta función devuelve dos ficheros, un fichero “.JSON” y un fichero “.BIN”. El primer de estos ficheros contiene información sobre la arquitectura de la red y los parámetros de entrenamiento, el segundo fichero contiene información sobre los datos que ha usado para entrenarse y como transfiere los pesos a las diferentes neuronas.

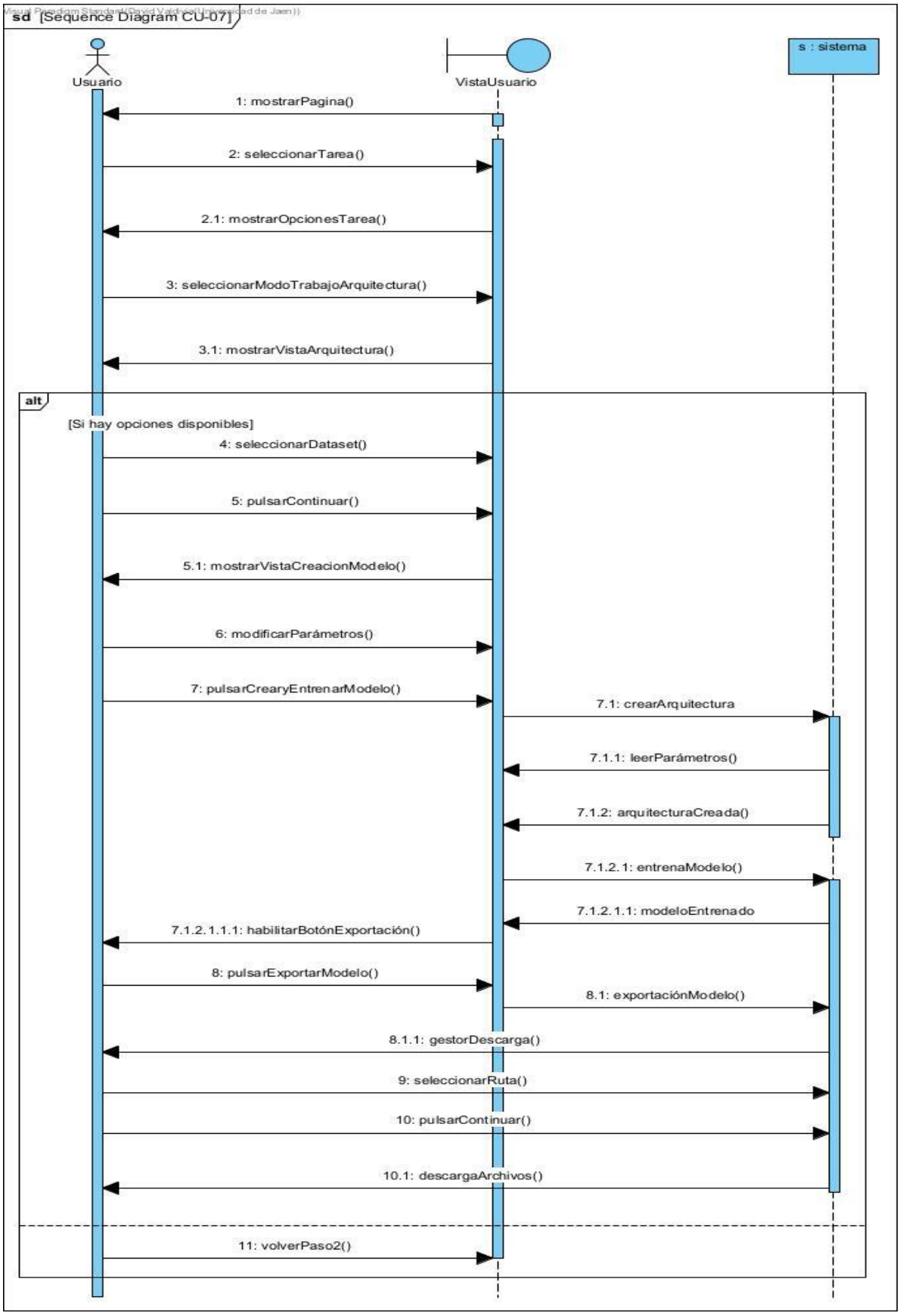

Figura 24: Diagrama de secuencia del caso de uso CU-07

La aplicación permite al usuario modificar parámetros a una arquitectura de ejemplo para ver y explorar los miles de posibilidades que nos brinda la librería de Tensorflow.js. Partiendo del caso de uso 3 la interfaz carga una nueva vista, en esta vista nos encontramos con una descripción del dataSet elegido, parámetros generarles de la red para su entrenamiento y un editor de capas que conforman la red (Figura 25).

Los parámetros generales son utilizados para el entrenamiento de la red, determinan el número de iteraciones, la tasa de entrenamiento y aprendizaje que se va a usar, la función de optimización para el entrenamiento, la función de pérdida y la función de métrica.

Los parámetros para el editor de capas dependen del tipo de capa que seleccionemos, los parámetros principales son número de neuronas por capa y función de activación de la capa. También se le permite al usuario añadir más capas del tipo que desee de forma que el usuario puede explorar cómo reacciona un modelo a diferentes parámetros.

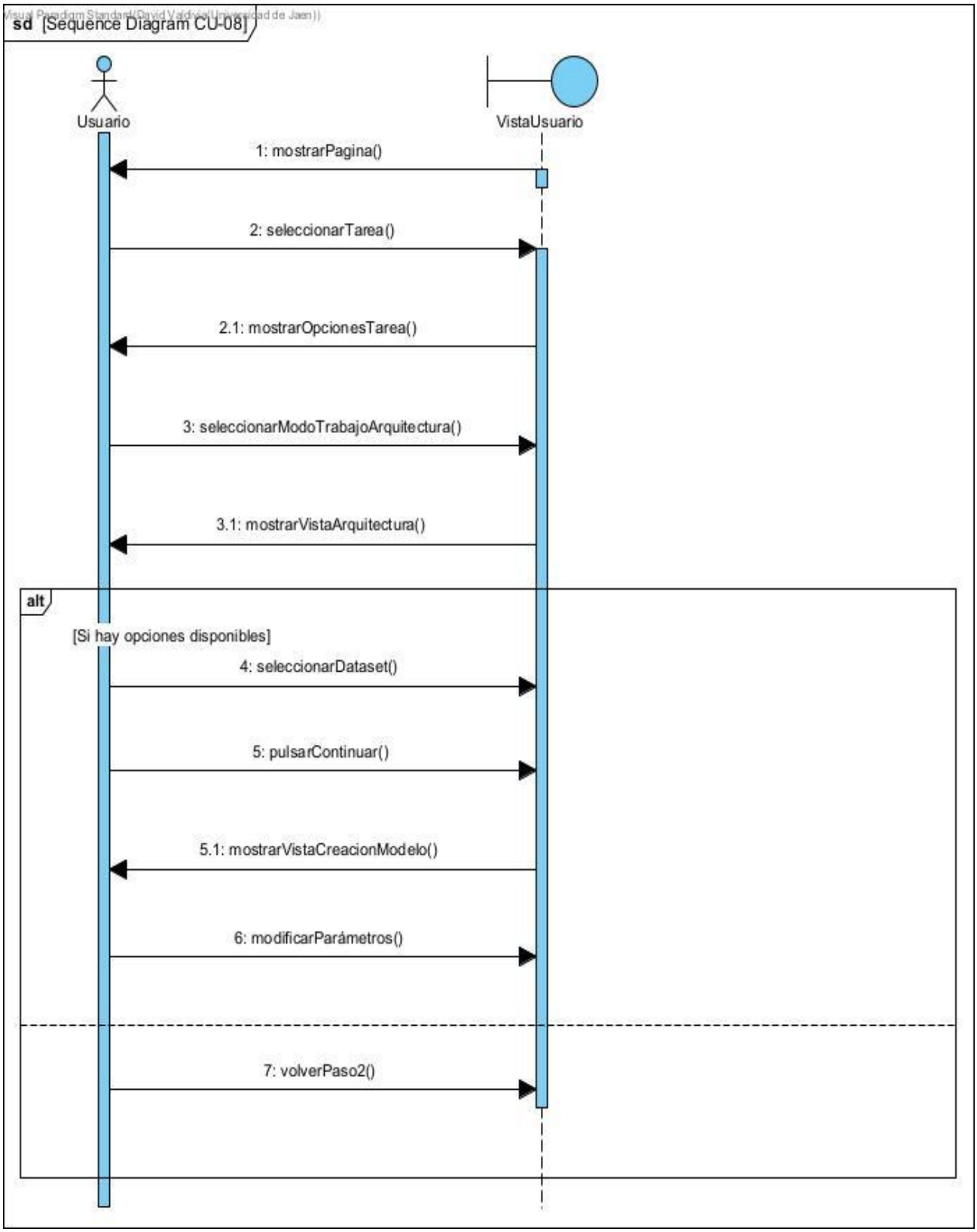

Figura 25: Diagrama de secuencia del caso de uso CU-08

Cuando el usuario ya ha modificado todos los parámetros que desea se inicia el proceso de entrenamiento (Figura 26). El usuario deberá de pulsar el botón “crear y entrenar modelo” y el sistema se encarga de leer todos los formularios, crear la arquitectura y entrenar el modelo con el data set seleccionado por el usuario. Si ha ocurrido algún problema al entrenar el modelo se notifica al usuario, este deberá de revisar los parámetros que ha puesto ya que probablemente la selección que ha hecho es incompatible. Si el modelo se ha entrenado correctamente, ya está listo para su evaluación.

Adicionalmente se ha agregado mediante una librería una serie de gráficas en las que podemos ver la pérdida y el acierto del modelo según avanza el entrenamiento, de forma que vemos la evolución.

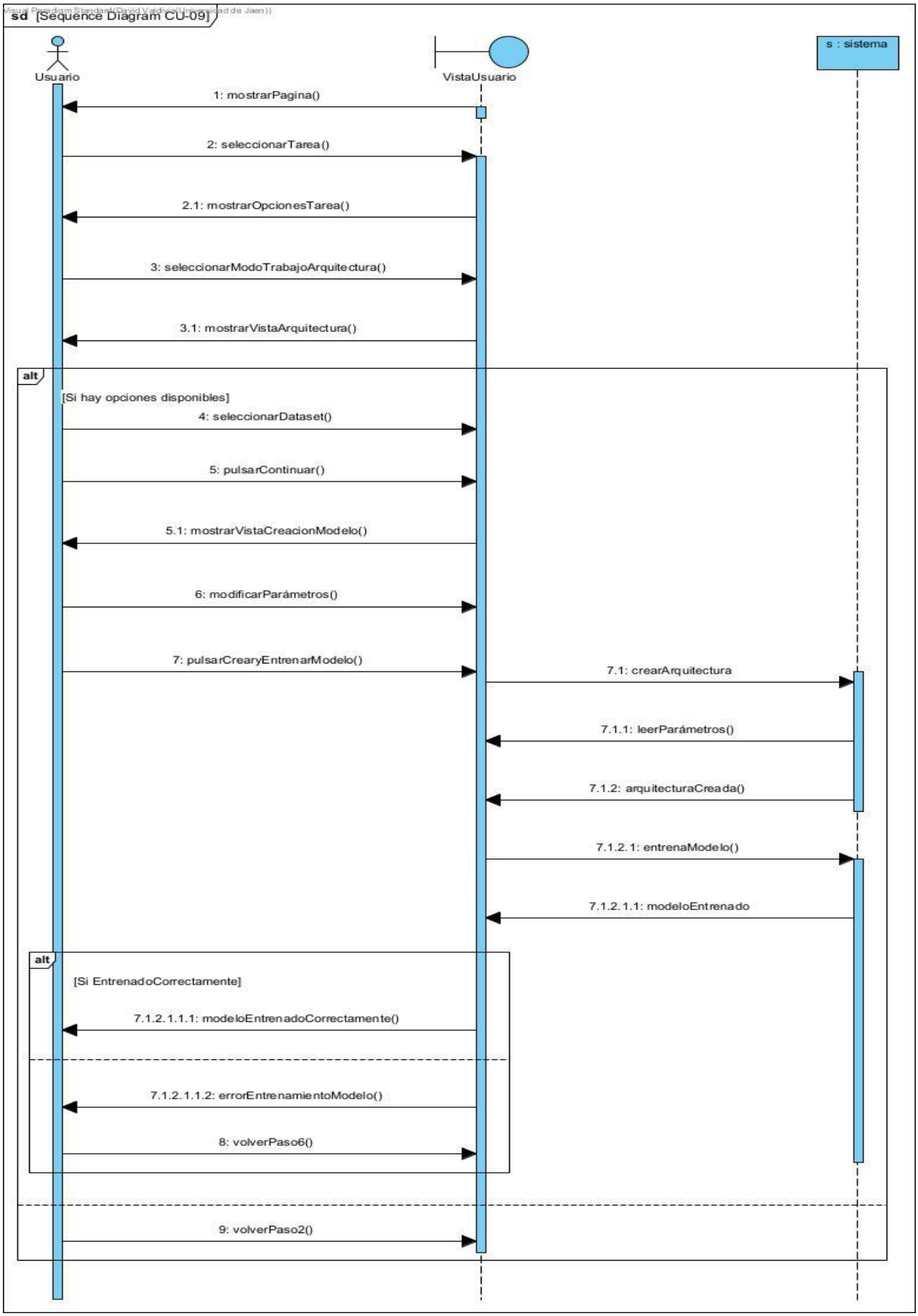

Figura 26: Diagrama de secuencia del caso de uso CU-09

La evaluación del modelo es muy sencilla de cara al usuario. Una vez que se ha entrenado la red correctamente el usuario puede completar el formulario final. Dependiendo de la tarea inicial elegida este formulario puede cambiar. Si se elige la tarea de clasificación clásica, el formulario contará con un campo donde se deben de introducir todos los parámetros separados por punto y coma (“;”), sin embargo, si elige clasificación de imágenes tendrá la opción de activar la webcam de su equipo o subir una imagen.

Después como se ve en la Figura 27 simplemente el usuario pulsa el botón “evaluar modelo” y el sistema se encarga de obtener la información del formulario de entrada y pasarlo a la red para predecir un valor. Si la evaluación ha sido satisfactoria la interfaz muestra una alerta con el resultado de la misma, en caso contrario muestra por qué no se ha completado la evaluación del modelo.

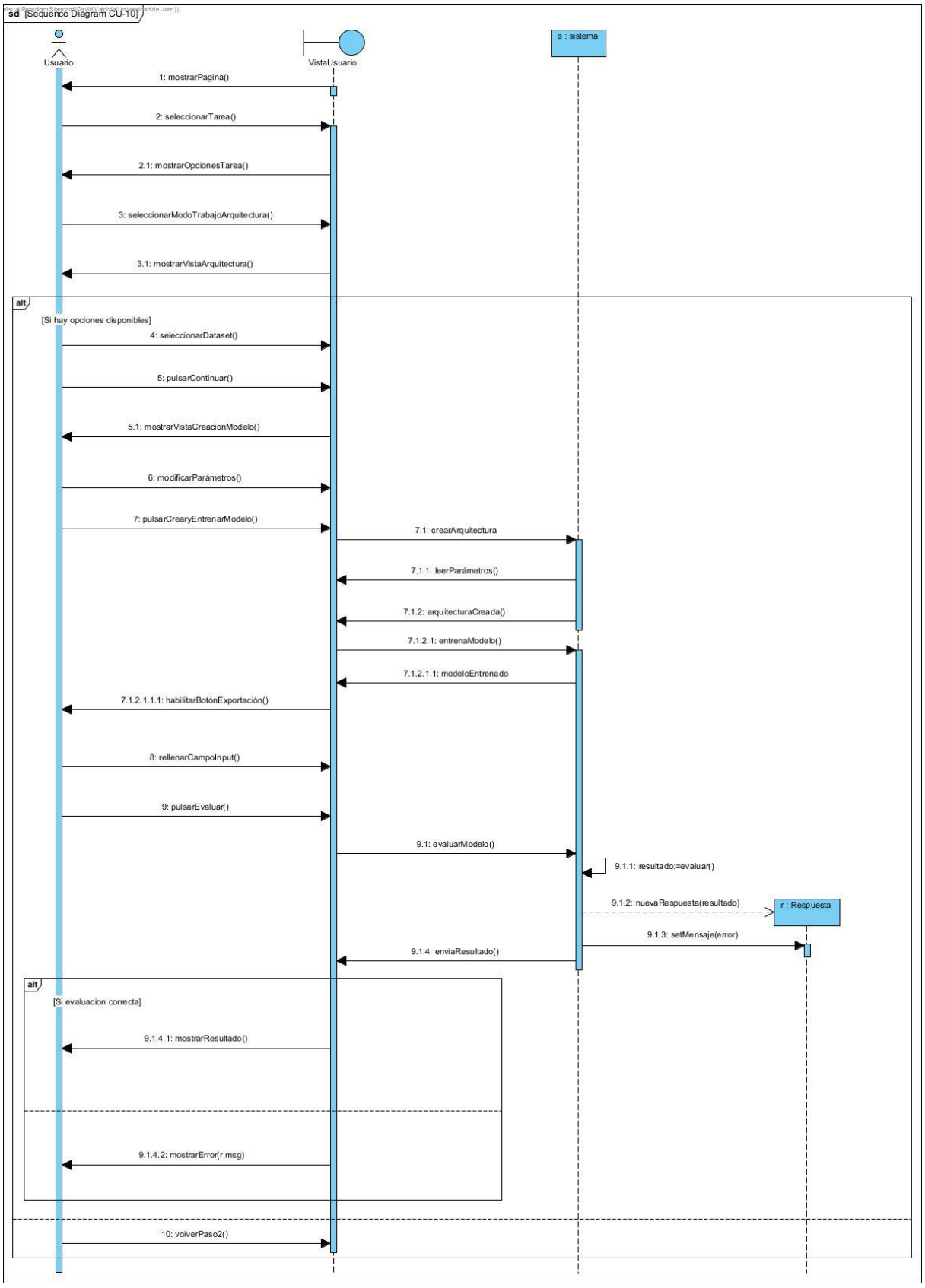

Figura 27: Diagrama de secuencia del caso de uso CU-10

4.2. Diagramas de clases.

Una vez modelados los diagramas de secuencia se genera el diagrama de clases (Figura 28). En este diagrama podemos observar la clase VistaUsuario, esta clase se refiere a la interfaz de usuario que representa el framework, en este caso React.js, que se va a utilizar.

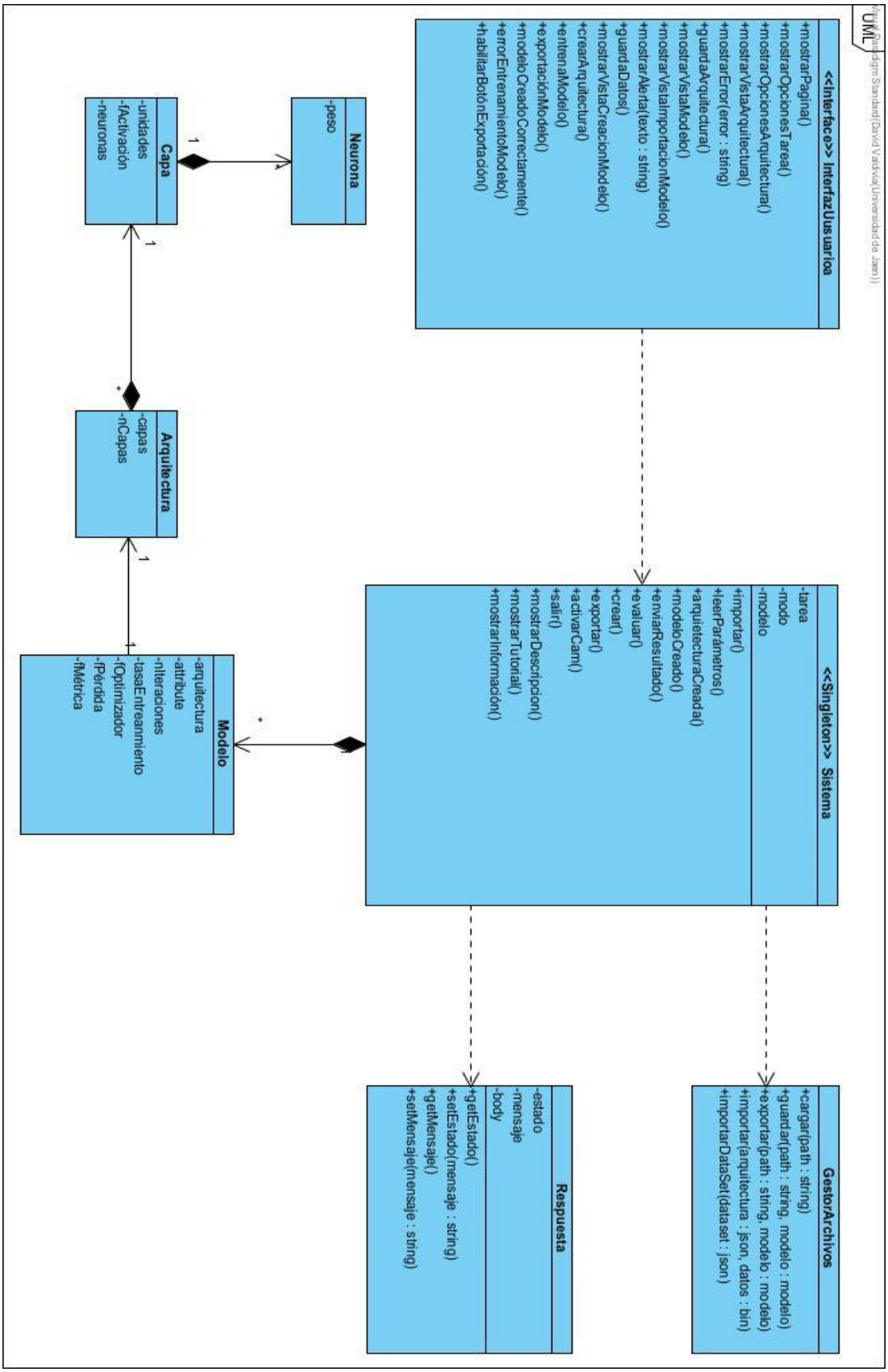
UML
<<Interface>> InterfazUsuario
+mostrarPagina()
+mostrarOpcionesTarea()
+mostrarOpcionesArquitectura()
+mostrarVistaArquitectura()
+mostrarError(error : string)
+guardaArquitectura()
+mostrarVistaModelo()
+mostrarVistaImportacionModelo()
+mostrarAlerta(texto : string)
+guardaDatos()
+mostrarVistaCreacionModelo()
+crearArquitectura()
+entrenaModelo()
+exportaciónModelo()
+modeloCreadoCorrectamente()
+errorEntrenamientoModelo()
+habilitarBotónExportación()
<<Singleton>> Sistema
-tarea
-modo
-modelo
+importar()
+leerParámetros()
+arquietecturaCreada()
+modeloCreado()
+enviarResultado()
+evaluar()
+crear()
+exportar()
+activarCam()
+salir()
+mostrarDescripcion()
+mostrarTutorial()
+mostrarInformación()
GestorArchivos
+cargar(path : string)
+guardar(path : string, modelo : modelo)
+exportar(path : string, modelo : modelo)
+importar(arquitectura : json, datos : bin)
+importarDataSet(dataset : json)
Respuesta
-estado
-mensaje
-body
+getEstado()
+setEstado(mensaje : string)
+getMensaje()
+setMensaje(mensaje : string)
Neurona
-peso
Capa
-unidades
-fActivación
-neuronas
Arquitectura
-capas
-nCapas
Modelo
-arquitectura
-atributte
-nIteraciones
-tasaEntreanmiento
-fOptimizador
-fPérdida
-fMétrica
1
1
1
1
1
*
*
*

Figura 28: Diagrama de clases.

5. DESARROLLO.

Esta sección va a abordar todo el proceso de desarrollo de software para la realización del **TFG** .

La estrategia de desarrollo que se va a seguir es un modelo incremental, como se ha mencionado en la sección anterior (véase la sección 2.5.2). El proceso de desarrollo se dividirá en iteraciones cuyo resultado es un incremento, la sucesión de las iteraciones nos dará el producto final.

5.1. Descripción de la solución propuesta

La solución propuesta será una aplicación web donde el usuario puede aprender a ejecutar y modificar diferentes modelos de aprendizaje automático. Además, podrá importar arquitecturas y modelos y exportar modelos.

5.2. Estudio de alternativas y viabilidad.

Este proyecto ha tenido varios meses de desarrollo, durante los mismo la idea del proyecto ha ido madurando en diferentes aspectos. Al inicio se buscaba realizar un editor de código para modelos de aprendizaje automático, tras realizar un primer incremento de la idea, se optó por simplificar la interfaz de cara al usuario ya que la aplicación solamente estaba dirigida para aquella gente que ya dispone de conocimientos en el campo del machine learning de forma que se quedarían al margen una gran cantidad de usuarios.

Tras pensar un poco la nueva idea se escogió realizar una aplicación más accesible y amigable de cara a un público sin conocimientos en el campo de la inteligencia artificial, incluso sin conocimientos en programación.

De esta forma se llegó a la idea que conforma el núcleo donde se ha construido esta aplicación. Una web dirigida a todos los públicos, con diferentes tipos de modelos de aprendizaje automático donde el usuario puede editar una arquitectura para intentar mejorarla, evaluar modelos proporcionados por la aplicación para que el

usuario pueda visualizar el potencial que tiene esta tecnología, en el caso de que el usuario disponga de conocimientos y haya desarrollado un modelo o una arquitectura por su cuenta, podrá importarla si ningún problema. También se pueden exportar los modelos creados por el usuario dentro de la aplicación Web.

En definitiva, se eligió esta opción porque agrega una parte divulgativa dando lugar a inspirar a gente y que se animen a aprender acerca de esta materia manteniendo funcionalidades de la idea original.

5.3. Bibliotecas VSCode

Como se ha explicado en 2.1 se usará Tensorflow.js, React.js y VSCode para el desarrollo del proyecto, también se usará GitHub para el almacenamiento del mismo en la nube y brindarnos la posibilidad de trabajar desde diferentes equipos.

Para la realización del proyecto se han usado librerías externas como apoyo. A continuación, se va a hacer un resumen de todas estas librerías para comprender la función que realiza cada una dentro de la aplicación.

- Emotion[14]: librería para escribir estilos **CSS** con JavaScript. Se han usado las siguientes dependencias: *"@emotion/react", "@emotion/styled*".
- Material UI[15]: librería de componentes React que proporciona robustez y estabilidad a los diseños web. Junto con esta librería van estas dependencias: *"@material-ui/core" y "@mui/system*".
- TensorFlow.js[16]: librería principal de este proyecto. Es una librería desarrollada por Google la cual nos permite ejecutar y crear modelos de aprendizaje automático desde el navegador con la ayuda del lenguaje JavaScript. De su instalación depende bastantes dependencias: *"@tensorflow-models/face-detection", "@tensorflow-models/face-landmarks-detection", "@tensorflow-models/pose-detection", "@tensorflow/tfjs" y "@tensorflow/tfjs-vis".*

[14] https://emotion.sh/docs/introduction
[15] https://mui.com/
[16] https://www.tensorflow.org/js/?hl=es

- Testing Library[17]: ayuda a probar los componentes de la interfaz de usuario. Los paquetes que cuelgan de esta instalación son: *"@testing-library/jest-dom", "@testing-library/react" y "@testing-library/user-event*".
- Bootstrap y react-bootsrap[18]: al igual que material-ui Bootstrap es una librería de componentes para el diseño web. Se han usado las siguientes dependencias: “Bootstrap” y “react-bootstrap”.
- Brace[19]: forma parte de la librería “browsery” la cual permite juntar todas las llamadas a la aplicación en una única etiqueta <script>
- Codemirror[20]: como se ha mencionado la idea inicial del proyecto era crear un editor de código para la craeción de modelos. Esta librería es un editor de texto en JavaScript. Aunque no se use para la apliación final, se ha mantenido en el proyecto para en el futuro aumentar la funcionalidad de la aplicación.
- React-codemirror2[21]: adaptación de la librería de JavaScript “codemirror” en componentes para React.
- React-dom[22]: proporciona métodos específicos del DOM de React
- React-icons[23]: librería de iconos para react con importación ES6 de forma que puedes importar solamente el icono que necesitas.
- react-router[24]: librería usada para el enrutamiento de las diferentes vistas dentro del entorno de React. Recordamos que React forma parte de los frameworks de “single page view” (Microsoft, 2022). Gracias a esta librería podemos crear varias rutas sin dejar de tener las ventajas de la single page view. Se han usado “react-router” y “react-router-dom”.
- react-scripts: dependencia instalada por defecto al crear un proyecto de React mediante “Create React App”[25].
- vis-network[26]: librería para mostrar gráficos de redes usando vis.js. Se ha utilizado para mostrar de forma sencilla la estructura de las arquitecturas

[17] https://testing-library.com/docs/
[18] https://getbootstrap.com/ https://react-bootstrap.github.io/
[19] https://github.com/browserify/browserify https://github.com/thlorenz/brace
[20] https://codemirror.net/
[21] https://github.com/scniro/react-codemirror2#readme
[22] https://es.reactjs.org/docs/react-dom.html
[23] https://react-icons.github.io/react-icons/
[24] https://reactrouter.com/
[25] https://create-react-app.dev/
[26] https://github.com/crubier/react-graph-vis#readme

y poder modificarlas de forma sencilla. Con la instalación de esta librería se incluyen “vis-network” y “react-vis-network-graph”

- react-webcam[27]: para algunos de los modelos utilizados por la aplicación se puede usar la webcam como entrada a la red, gracias a esta librería podemos incluir el video capturado por la webcam de forma sencilla.
- web-vitals[28]: librería incluida con la creación del proyecto mediante “Create React App”, esta librería permite obtener información acerca del rendimiento de la aplicación.
- sweetalert2[29]: librería que nos permite cambiar las alertas del navegador. En este proyecto las alertas del navegador son usada para notificar al usuario la predicción que han realizado los modelos, gracias a esta dependencia podremos darle a la alerta un formato más actual y más bonito.

5.4. Incrementos realizados.

Como se ha visto en el punto 2.5, la metodología que se va a usar para completar este proyecto es por incrementos. La aplicación se ha dividido en un total de cuatro incrementos, todos ellos ampliando la funcionalidad de la aplicación.

El primer incremento ha estado formado por la creación del proyecto de React, instalación de algunas librerías auxiliares comentadas en el apartado anterior y dotar a la aplicación web de una estructura básica. Se han definido los menús y la interfaz básica. Es la base del proyecto, ya que a partir de aquí nacen las diferentes vistas de la web y está incluida toda la lógica de rutas utilizadas con la ayuda de la librería “react-router-dom”. Contamos con un menú anidado en el cual de una opción de divide en varias conforme vas bajando los niveles.

En el segundo incremento se ha incluido el desarrollo de la tarea de clasificación clásica. Las funcionalidades incluidas en este incremento las vamos a dividir en dos partes.

[27] https://github.com/mozmorris/react-webcam
[28] https://github.com/GoogleChrome/web-vitals#readme
[29] https://sweetalert2.github.io/

La primera parte está relacionada con la creación y edición de arquitecturas y modelos de aprendizaje automático. El usuario tiene la oportunidad de cargar su propia arquitectura en un formato .json, este json debe de tener una estructura específica, esta estructura es proporcionada por la api de tensorflow.js concretamente la función Model.save() utilizada para la exportación de modelos genera un archivo json, ese es el formato que debe de tener.

Además de poder elegir una arquitectura el usuario puede seleccionar qué dataSet se va a utilizar para crear y entrenar a la red. Mediante un desplegable se le muestran varios datasets al usuario. Dependiendo de la selección el modelo se entrenará en base a esos datos y por consiguiente la predicción que realice el modelo la hará en torno a esos datos iniciales de entrenamiento.

Ahora es momento de poder crear la arquitectura y entrenar al modelo, en este incremento se ha incluido un formulario con diferentes parámetros de la arquitectura, estos parámetros definen el número de capas que tiene la red y como están formadas y una serie de valores y funciones explicadas en el apartado 2.3.

Después de la modificación de la red prosigue el entrenamiento de la misma, mediante un botón que pulsa el usuario, la red comienza a entrenarse con los parámetros introducidos y con el dataSet seleccionado. Es importante tener en cuenta que este proyecto no incorpora un servicio de Back-end por lo tanto toda la ejecución y procesamiento de cualquier dato que realice la web se hace en el equipo del cliente, el que está visualizando la web. Esto nos proporciona una serie de ventajas y desventajas, las ventajas son que no se depende de un Back-end por lo que la aplicación es mucho más económica y fácil de mantener, sin embargo, la capacidad de procesamiento está limitada por las características de la máquina del cliente y por las funcionalidades que incorporan las librerías de JavaScript. En un servicio de Back-end podemos utilizar el lenguaje de programación Python el cual cuenta con una serie de librerías bastantes más potentes para crear, entrenar y ejecutar modelos de aprendizaje automático. Tensorflow.js que es la librería que usamos para este proyecto y que nos permite ejecutar los modelos desde un Front-end, es una adaptación de la librería original Tensorflow dirigida a Python.

Por último, pero no menos importante, tras el correcto entrenamiento de la red se ha incorporado la última funcionalidad del segundo incremento, la evaluación del modelo. Para evaluar el modelo se ha decidido incorporar un formulario donde el usuario puede introducir los datos iniciales separados por punto y coma y mediante el procesamiento del lenguaje natural el propio procesamiento de la web se encarga de transformar esa entrada de texto a valores que la red pueda entender y dárselo a la red para que haga una predicción. Cuando la red acaba de realizar su predicción el resultado de la misma se mostrará al usuario mediante una alerta en el navegador.

En la segunda parte que formaba el segundo incremento nos encontramos con la ejecución de modelos ya entrenados para que el usuario pueda visualizar el funcionamiento y potencial de una red que ya ha sido entrenada de forma óptima. En el entrenamiento de redes neuronales no hay un entrenamiento de formas más correcta o incorrecta, simplemente hay redes que tienen una arquitectura más óptima para cada tipo de problema y por consiguiente esta red será mejor (dará mejores predicciones) que otras redes.

Para la ejecución de modelos ya entrenados el usuario debe de elegir entre diferentes modelos en un desplegable, tras elegir uno se cargará una nueva vista donde el modelo ya ha sido cargado en memoria y donde el usuario debe de rellenar un formulario para incorporar los valores de entrada para la evaluación de la red. Tras la evaluación de igual forma que se ha comentado anteriormente el resultado se muestra en forma de alerta en el navegador web.

En el tercer incremento se incorporó la tarea de la clasificación de imágenes. La clasificación de imágenes al igual que la clasificación clásica nos sirve para determinar la clase a la que pertenece el dependiente según una o más variables independientes. En este caso la variable de entrada es una imagen.

Este tercer incremento tiene una estructura similar al segundo incremento, al incorporar una nueva tarea esta se divide en ejecución de modelos pre-entrenados y en la creación y entrenamiento de nuevas redes creadas por el usuario.

En la creación y entrenamiento de modelos se añadieron algunas funcionalidades para adaptar la web a las características de los modelos enfocados a

la clasificación de imágenes. Se ha agregado un gráfico de nodos que representan cada una de las capas de la red, de forma que a simple vista puedes ver la estructura de la misma. Para modificar una de estas capas simplemente el usuario tiene que pulsar en la capa que desea modificar (el nodo del gráfico) y se recargará el formulario de la capa con sus datos rellenos y listo para que el usuario pueda modificarlos. De igual forma cuando el usuario desea introducir una nueva capa, simplemente pulsa el botón de añadir capa y automáticamente aparecerá otro nodo en el gráfico y se cargará su formulario.

Además del gráfico de nodos para la edición de las capas de la red, se ha incorporado un sistema de gráficas que nos informa como avanza tanto la función de pérdida como la precisión de la red durante el entrenamiento. Para su visualización simplemente vale con pulsar el botón “Entrenar modelo” y aparecerá un panel en el margen derecho que nos arroja distinta información de gran utilidad. Primero nos muestra la estructura final de la arquitectura, con la forma que tienen cada una de las capas de la red y después nos muestra las gráficas de cómo avanza la pérdida y la precisión a través de las diferentes etapas del entrenamiento.

Por último, en cuanto a la evaluación del modelo, al tratarse de clasificación de imágenes el formulario de entrada para evaluar el modelo ha cambiado. Dependiendo del dataSet seleccionada variarán el formulario de entrada, pero las opciones que abarcan son: la subida de una imagen desde el equipo del usuario y dibujar números a través de un canvas incorporado.

En cuanto a la segunda opción del tercer incremento no hay grandes diferencias respecto al segundo incremento, simplemente en la entrada de los datos, pero el resto de funcionalidades que había en el segundo incremento funcionan exactamente igual.

Para el final tenemos a el **cuarto** y último **incremento.** Para este incremento se ha agregado la tarea de detección de objetos. A diferencia de la detección de imágenes que analizan toda la imagen para asignarle una categoría, la detección de objetos analiza partes de una imagen para clasificarla, organiza elementos de acuerdo a sus diferencias y semejanzas.

Este cuarto incremento solo consta con la opción de ejecutar modelo pre-entrenados debido a que estas redes adquieren un nivel medio alto de complejidad y están formadas por una cantidad considerable de capas, de forma que se hace complicado que un usuario pueda realizar una red con estas características desde cero sin tener conocimientos previos y agruparlo todo en una interfaz de usuario amigable.

A la hora de evaluar los modelos se ha agregado una nueva opción, el uso de la webcam. Se ha conseguido ejecutar los modelos en tiempo real enviando como datos de entrada la señal de video de la webcam de forma que obtenemos los resultados prácticamente al instante. Esta funcionalidad es muy interesante de cara al usuario ya que es muy llamativa, porque se obtienen los resultados rápidamente y el usuario ve con facilidad lo que hace el modelo, además los modelos empleados están a la orden del día, por ejemplo, hay uno que marca con una serie de puntos una malla sobre la cara, este modelo es muy usado en las redes sociales para colocar los famosos filtros sobre la cara y eso se lleva a cabo gracias a estos tipos de modelos.

6. CONCLUSIÓN Y TRABAJOS FUTURO.

Se ha construido una plataforma poco común con una gran capacidad divulgativa, capaz de facilitar el acceso a las redes neuronales a personas poco cualificadas en el área. Hoy en día se ha establecido la tecnología como forma de vida y esta avanza con paso acelerado, es por ello, que es muy importante una plataforma como la de este proyecto, para ayudar a las personas a evolucionar conforme lo hace la tecnología.

Se han desarrollado diferentes funcionalidades, como la creación de modelos de aprendizaje automático de forma totalmente dinámica, importación y exportación de modelos, creación de modelos a partir de una arquitectura y a partir de un dataset proporcionado por el usuario, evaluación de los modelos a través de texto, foto o vídeo, visualización de una gráfica de nodos, que representa la arquitectura del modelo que se está haciendo y visualización de diferentes gráficas durante el entrenamiento del modelo, donde se puede observar cómo evoluciona el modelo a través de las distintas etapas.

Se debe resaltar que debido a la limitación de tiempo que tienen este tipo de proyectos, se ha desarrollado todo de forma modular, con la idea de que el proyecto sea continuado, para incluir nuevas funcionalidades, mejorar las viejas y hacer de esta herramienta, la mejor herramienta de divulgación sobre redes neuronales y aprendizaje automático.

Respecto a las propuestas para siguientes trabajos están, renovar la interfaz gráfica de la aplicación, incluir más categorías de uso, incluir soporte para varios idiomas, realizar una wiki sobre cada uno de los conceptos que aparece y realizar un tutorial interactivo con los usuarios.

7. APÉNDICES

A continuación, se introduce documentación adicional de utilidad para comprender correctamente el proyecto.

7.1. Guía original del Trabajo Fin de Grado.

(Cód.: 21/22-2974) **Plataforma Web para el diseño y ejecución de modelos de aprendizaje profundo.**

Tutor del TFG**: ANTONIO JESÚS RIVERA RIVAS**

Modalidad: Proyecto de Ingeniería | Tipo: TFG Específico

Número máximo de estudiantes: 1 (0 asignados)

Idioma: Castellano

Segundo tutor del TFG: **MARÍA DOLORES PÉREZ GODOY**

La aparición del aprendizaje profundo supone un punto de inflexión dentro del campo del aprendizaje automático, desde el momento que permite abordar problemas en los que la inteligencia computacional clásica presenta serios problemas de eficiencia. Un ejemplo de estos problemas son los pertenecientes al área de la clasificación de imágenes o de la identificación de objetos en éstas. De otra parte, últimamente están apareciendo cada vez más lenguajes y herramientas que facilitan la programación de distintas aplicaciones que aprovechan las ventajas de este tipo de aprendizaje

El objetivo de este TFG es el diseño de una plataforma Web que permita el diseño y la ejecución de modelos de aprendizaje profundo en los clientes que se conecten a ella. Concretamente, los usuarios de esta plataforma podrán diseñar nuevos modelos de aprendizaje profundo, modificar algunos modelos predefinidos, entrenarlos y ejecutarlos.

Conocimientos Previos: No requiere

Objetivos del TFG*: Implementación de un módulo para el diseño de modelos de aprendizaje profundo que permita editar o modificar el código de los modelos

* Diseño de un módulo para el aprendizaje de modelos donde el usuario podrá configurar los parámetros de estos y entrenarlos

* Desarrollo de un módulo para la ejecución de los modelos ya entrenados en la máquina cliente

Metodología a Desarrollar: La metodología a seguir estará basada en la metodología genérica contemplada ingeniería de software para el desarrollo de aplicaciones y consta de los siguientes pasos:

- Análisis del problema
- Diseño de producto software
- Implementación
- Prueba

Documentos y Formatos de Entrega: Código y memoria en formato digital

8. MANUALES

A continuación, se muestran un manual de instalación de la aplicación en local y un manual de usuario para comprender el funcionamiento completo de la aplicación.

8.1. Manual de instalación

El manual de instalación refiere a un tutorial de cómo se debe de instalar la aplicación en un equipo para su ejecución o subida a producción. La aplicación es compatible con todos los sistemas operativos del mercado actualmente, se va a realizar una guía en profundidad del proceso de instalación en Windows y se va a notificar las diferencias que puede haber para instalarlo en el resto de sistemas operativos. Además, se muestran los requisitos mínimos a nivel de hardware que deben de tener las máquinas para el correcto funcionamiento de la aplicación.

Es importante tener en cuenta que la aplicación se puede lanzar de dos formas, en modo desarrollo y en modo producción. En la primera el proceso de lanzamiento es un poco más complicado y cuando lazamos el servidor nos muestra todos los errores que ocurren durante la ejecución. El segundo es el que se usa para instalar aplicaciones en los servidores web.

8.1.1. Requisitos mínimos

- Procesador: Core i3-6100 o equivalente.
- Memoria: 4 GB de RAM
- Gráficos: Intel HD Graphics 530
- Almacenamiento: 3 GB

8.1.2. Windows (modo desarrollo)

Para realizar la instalación en Windows, necesitamos la ayuda de algunos programas externos al proyecto para su ejecución.

Primero nos dirigimos a la web oficial de Node.js[30] (Figura 29) y descargamos la versión LTS para Windows.

Figura 29: Captura de pantalla de la página oficial de Node.js

Tras ello ejecutamos el archivo que hemos descargado con la extensión "**.MSI**" y se nos abrirá el proceso de instalación. El proceso es muy sencillo, simplemente se ha aceptar unos términos, seleccionar la ruta de instalación y escoger las características que se van a instalar (Figura 30). Para no tener ningún tipo de problema se recomienda instalar todas las funcionalidades.

[30] https://nodejs.org/es/

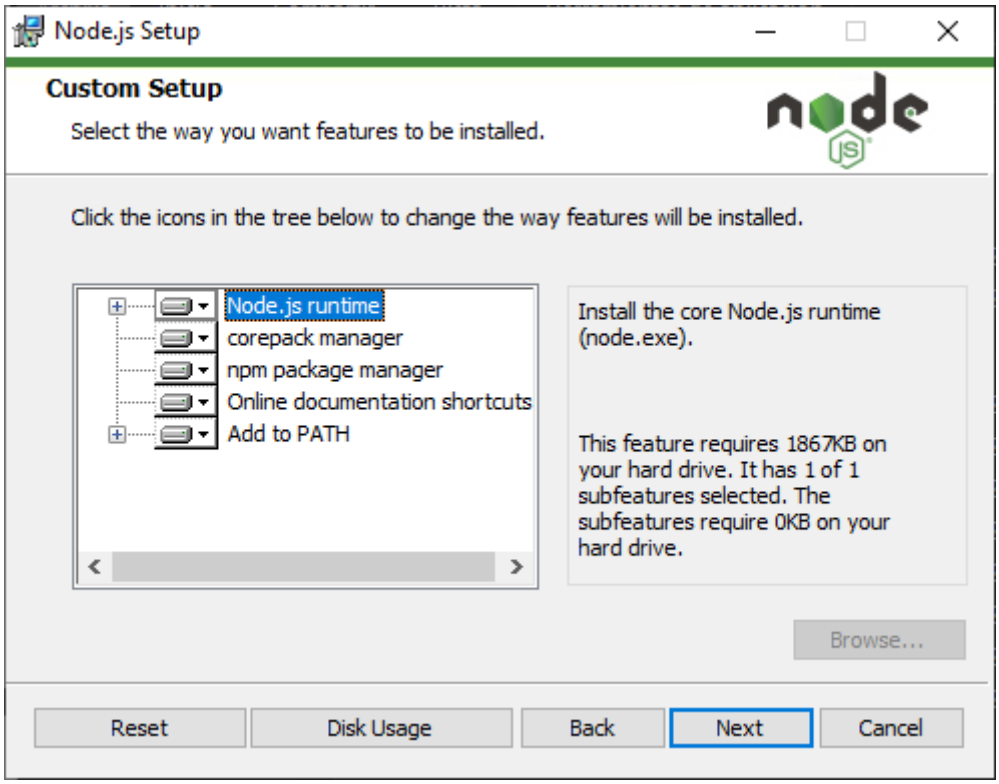

Figura 30: Proceso de instalación de Node.js

Ahora simplemente debemos de descomprimir la carpeta del proyecto en un lugar cómodo para nosotros, en el caso del tutorial se hace dentro de una carpeta en el escritorio del equipo (Figura 31).

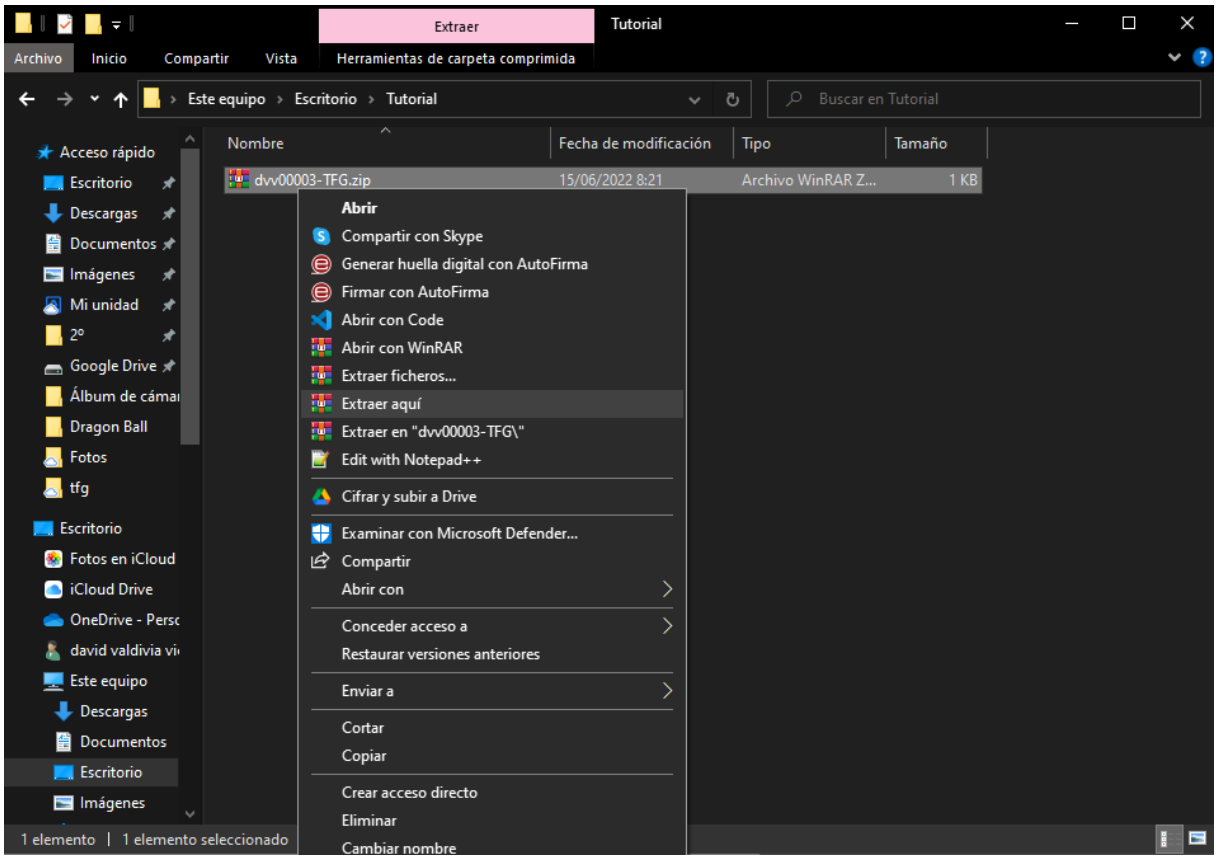

Figura 31: Descomprimimos el proyecto en el escritorio

Una vez se ha descomprimido nos quedará una carpeta. Ahora tenemos que acceder a esa carpeta desde el símbolo del sistema (CMD), para ello abrimos la CMD y navegamos hasta la carpeta que hemos descomprimido.

Ahora tenemos que introducir en la consola una serie de comandos, el primero de ellos es "*npm install*" (Figura 32), este comando se usa para instalar todas las dependencias que tiene el proyecto, de no hacerlo, al ejecutar el proyecto no da error ya que no puede encontrar las librerías a las que se hace referencia dentro del proyecto.

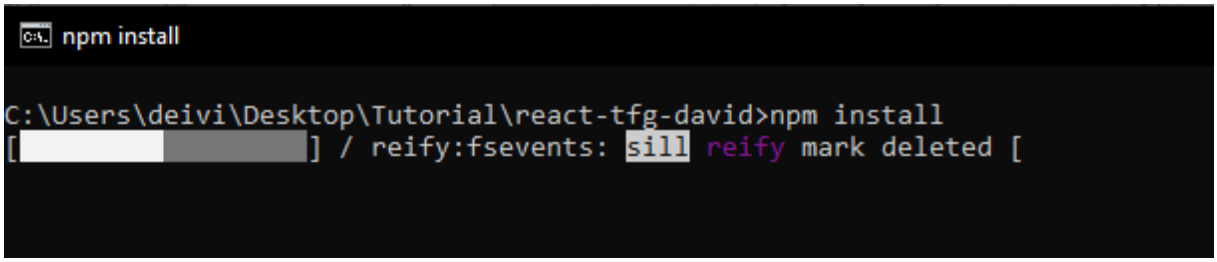

Figura 32: Ejecución del comando "npm install"

Debemos de esperar a que este proceso finalice (Figura 33 para continuar con el tutorial. Es posible que en un futuro al finalizar la ejecución de este comando nos aparezca una información acerca de algunos paquetes que son vulnerables. Esto es debido a que todas las librerías que se han usado para realizar este proyecto tienen una versión y siempre se va a instalar la misma versión de todas las librerías, aunque hayan aparecido nuevas versiones. Esta es una forma de asegurarse de que el proyecto puede ejecutarse en el tiempo, pero tiene un inconveniente y es que puede ser que algunos paquetes tengan alguna vulnerabilidad.

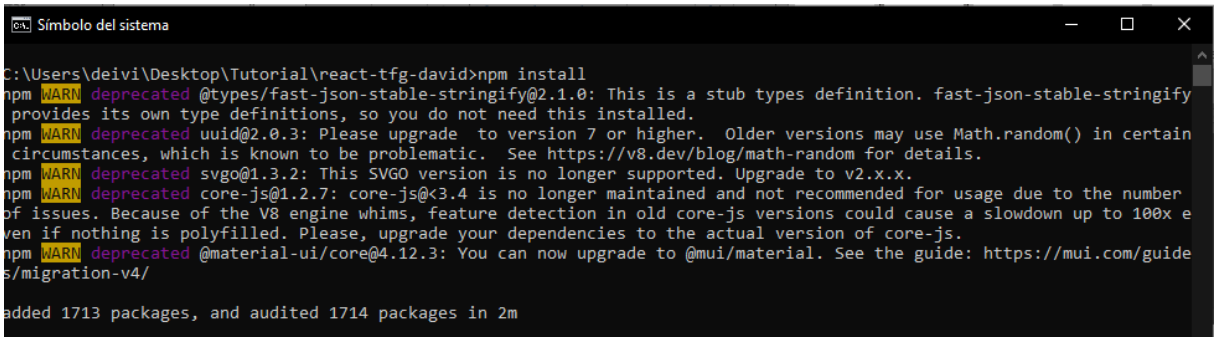

Figura 33: Ejecución de “npm install” ha finalizado

A continuación, debemos de introducir el comando “*npm start*” para lanzar la aplicación web y automáticamente se nos abrirá el navegador con la aplicación lista (Figura 34).

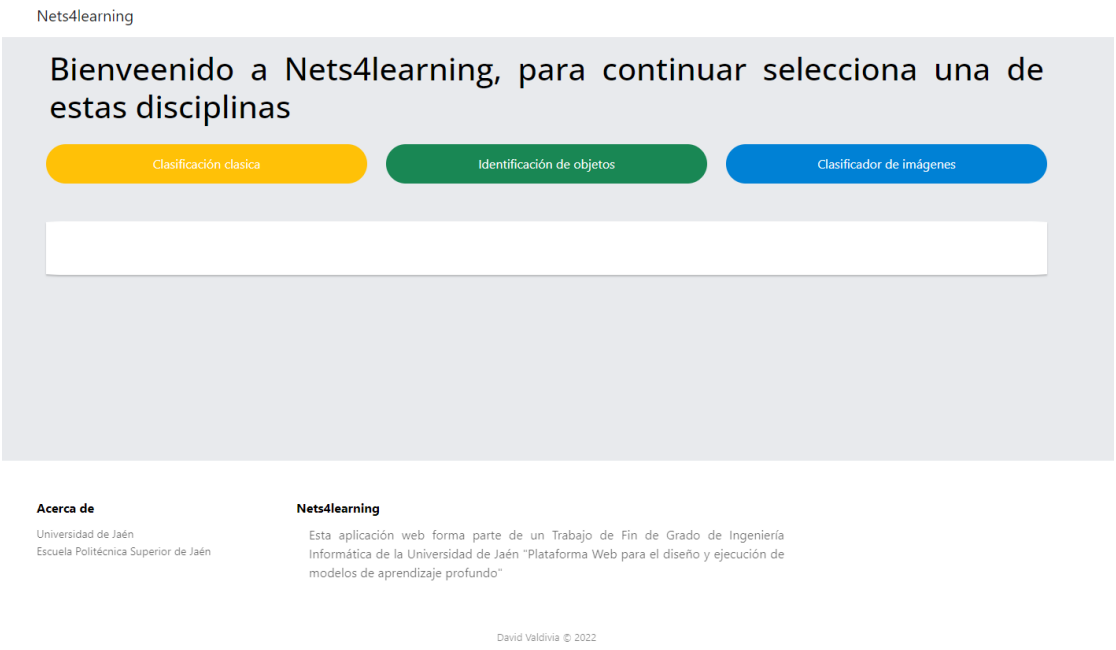

Figura 34: Aplicación web lanzada.

8.1.3. *MacOs (modo desarrollo)*

De igual forma necesitamos instalar Node.js, accedemos a su web oficial y seleccionamos la versión LTS. El proceso de instalación es muy sencillo, simplemente pulsamos continuar hasta finalizar el proceso (Figura 35).

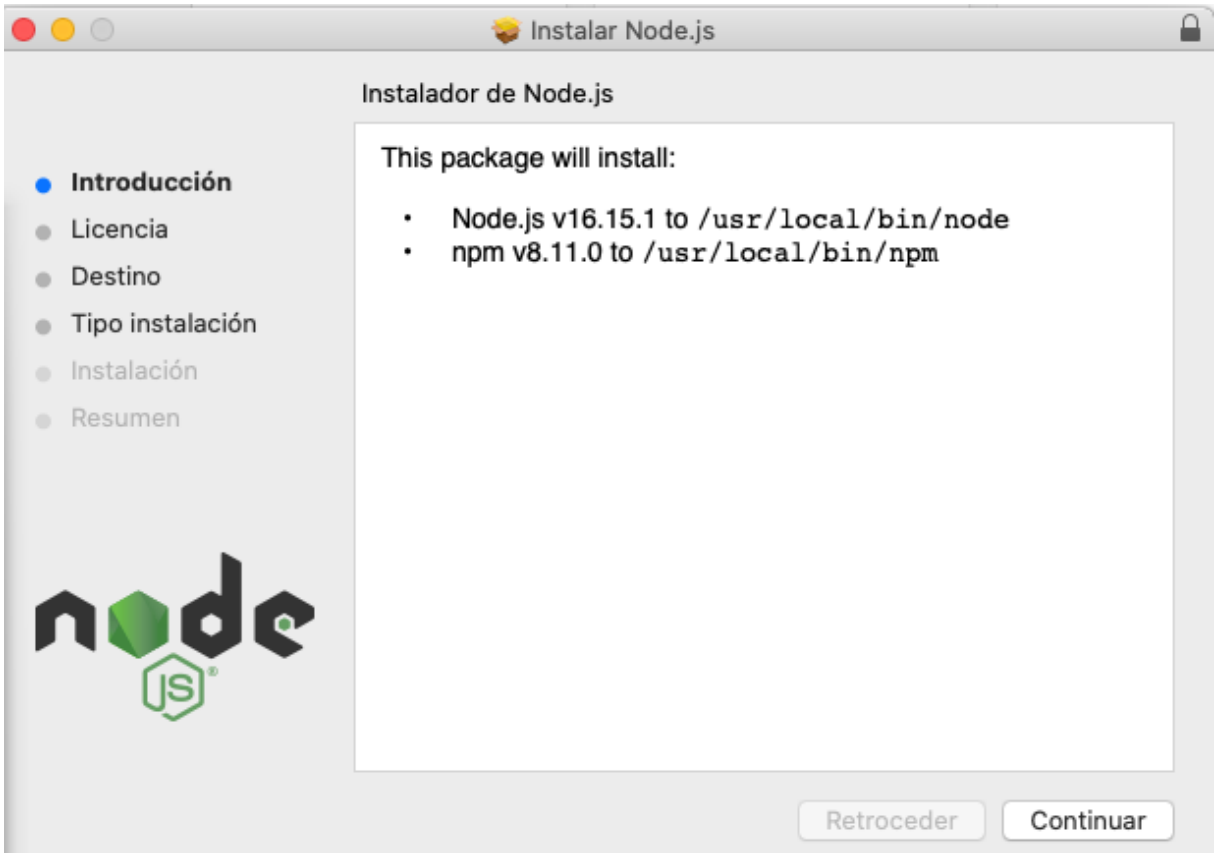

Figura 35: Instalación de Node.js en MacOs.

De igual forma que para Windows, descomprimimos la carpeta del proyecto, accedemos a ella mediante el terminal y ejecutamos los comandos "*npm install*" y "*npm start*", para descargar todas las dependencias y lanzar el servidor.

8.1.4. *Linux -Ubuntu (modo desarrollo)*

En Linux también necesitamos el entorno de Node.js. Nos dirigimos a su página web y navegando por la web nos dirigimos a descargar y a descargas usando un gestor de paquetes[31]. Ahora elegimos el sistema Linux que se esté usando. En mi

[31] https://nodejs.org/es/download/package-manager/#debian-and-ubuntu-based-linux-distributions-enterprise-linux-fedora-and-snap-packages

caso uso Ubuntu y los comandos de instalación son "*curl -fsSL https://deb.nodesource.com/setup_16.x | sudo -E bash –"y "sudo apt-get install -y nodejs*"

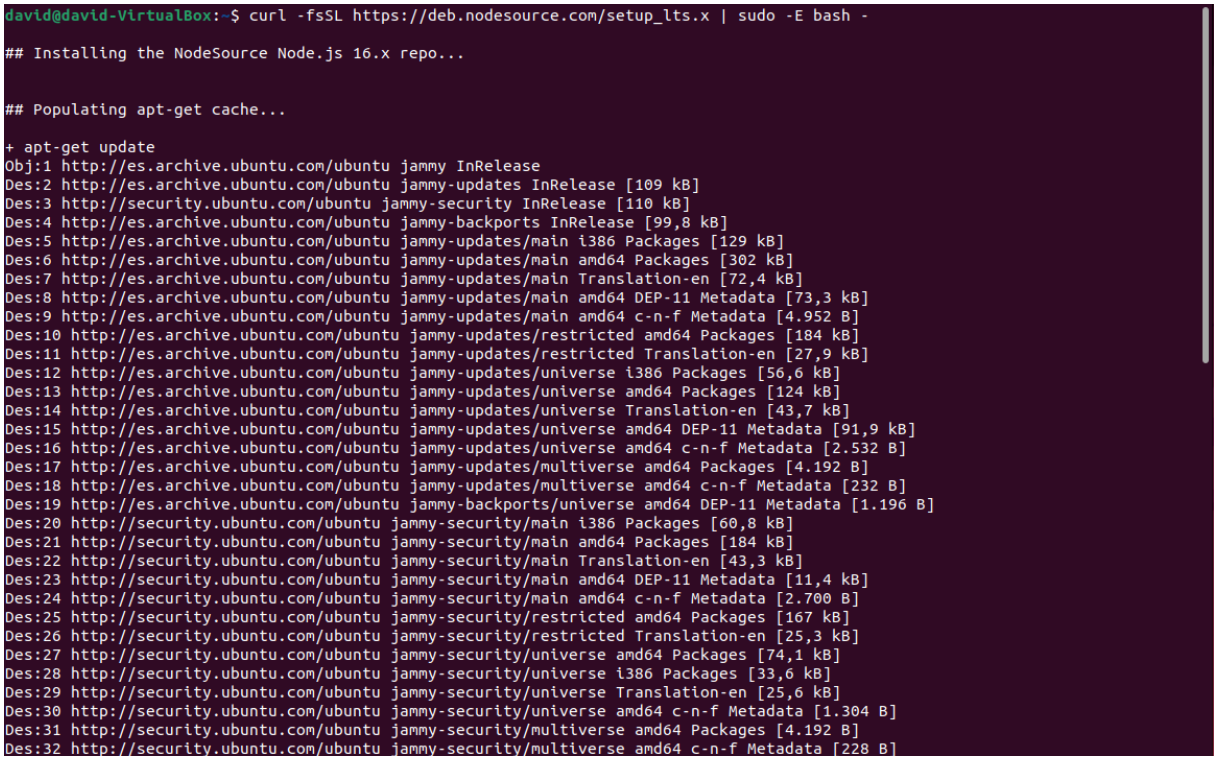

Figura 36: Instalación de Node.js (I)

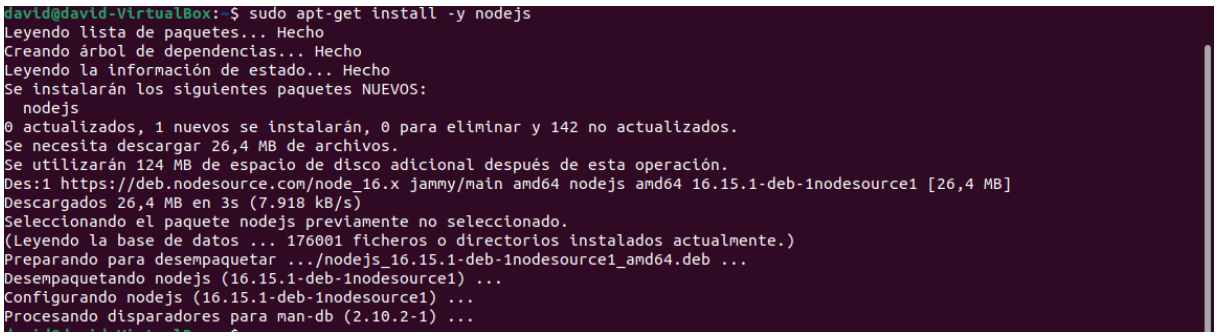

Figura 37: Instalación de Node.js (II)

Como se observa en las Figura 36 y Figura 37 mediante esos dos comandos se completa la instalación de Node.js.

Ahora de igual forma que en los anteriores **SSOO** descomprimimos la carpeta del proyecto y ejecutamos "*npm install*" y "*npm start"*

8.1.5. Modo producción

De forma similar al modo de desarrollo, descomprimimos el zip del proyecto en su versión de desarrollo. Una vez que hemos descomprimido la carpeta, accedemos a ella y escribimos estos comandos, "*npm install -g serve" y "serve -s build",* estos comandos instalan una librería de npm que permite ejecutar un servidor, y el segundo comando se encarga de lanzar este servidor (Figura 38)

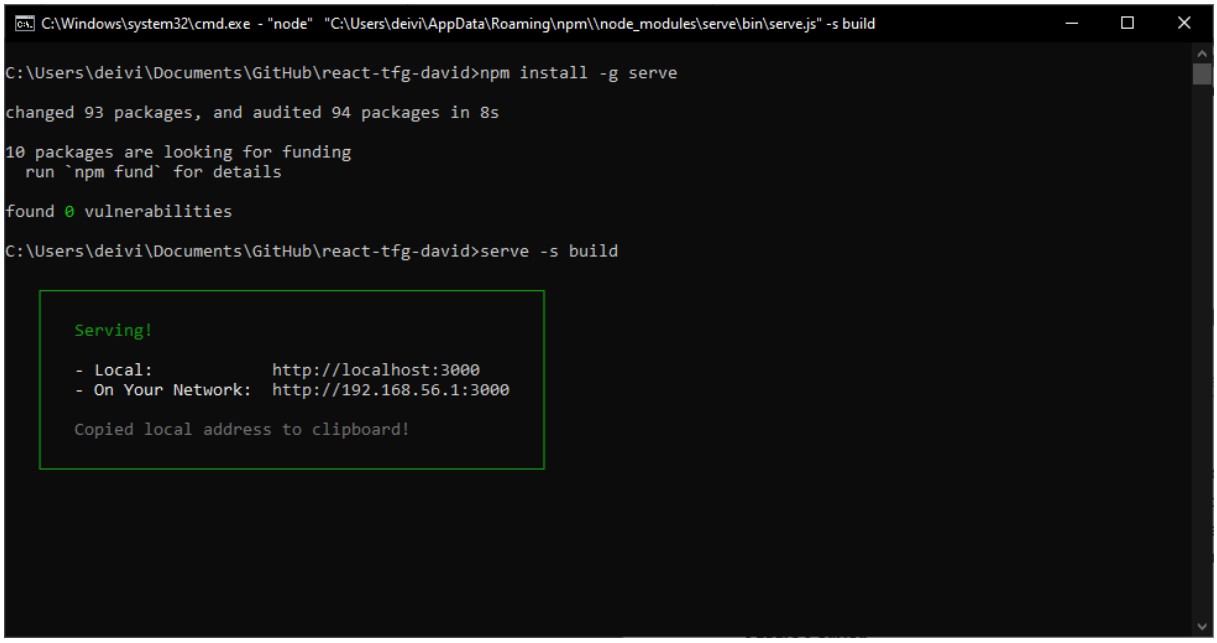

Figura 38: Lanzamiento del servidor de producción

Para lanzar la aplicación en modo producción se hace de igual modo en todos los sistemas operativos, de forma que es innecesaria una guía de cómo lanzar estos comandos desde los diferentes SSOO.

8.2. Manual de usuario.

El manual de usuario se va a realizar por comodidad desde un entorno Windows, no hay ninguna diferencia en la aplicación al lanzarla desde diferentes sistemas operativos.

8.2.1. Menú inicial

Cuando entramos en la web de la aplicación lo primero que vemos es una interfaz sencilla e intuitiva donde se ven tres botones de distinto color, cada uno de ellos nos hace referencia a un tipo de modelo de aprendizaje automático (Figura 39)

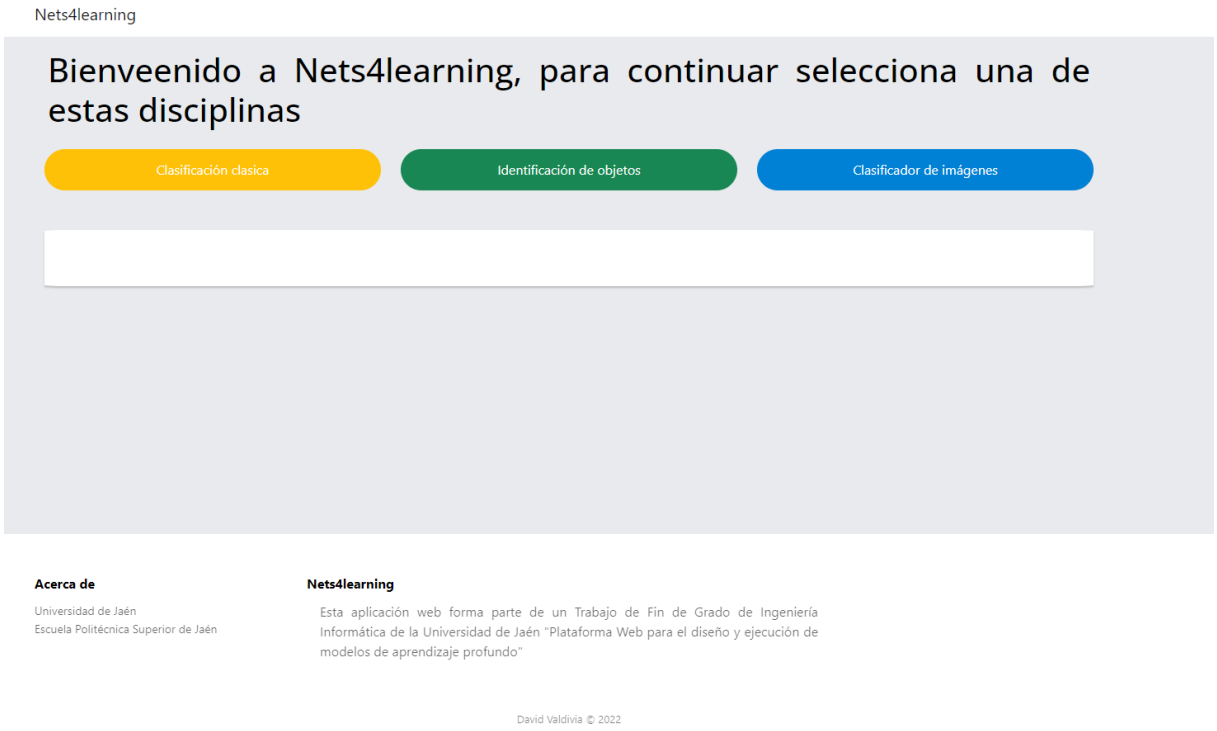

Figura 39: Interfaz principal de la aplicación

Si pulsamos sobre uno de estos botones nos mostrará una descripción de la disciplina que hemos seleccionado y una serie de botones Figura 40.

Figura 40: Interfaz de la aplicación tras pulsar un botón

Ahora visualizamos dos botones, “Modelo Pre-entrenado” y “Crear/Editar arquitectura”. El primero nos permite seleccionar un modelo que ya ha sido entrenado para poder evaluarlo, el segundo botón nos permite seleccionar un dataSet inicial y crear una arquitectura y un modelo desde cero.

8.2.2. Modelo Pre-entrenado

Si pulsamos sobre el botón “Modelo Pre-entrenado” veremos esta interfaz Figura 41.

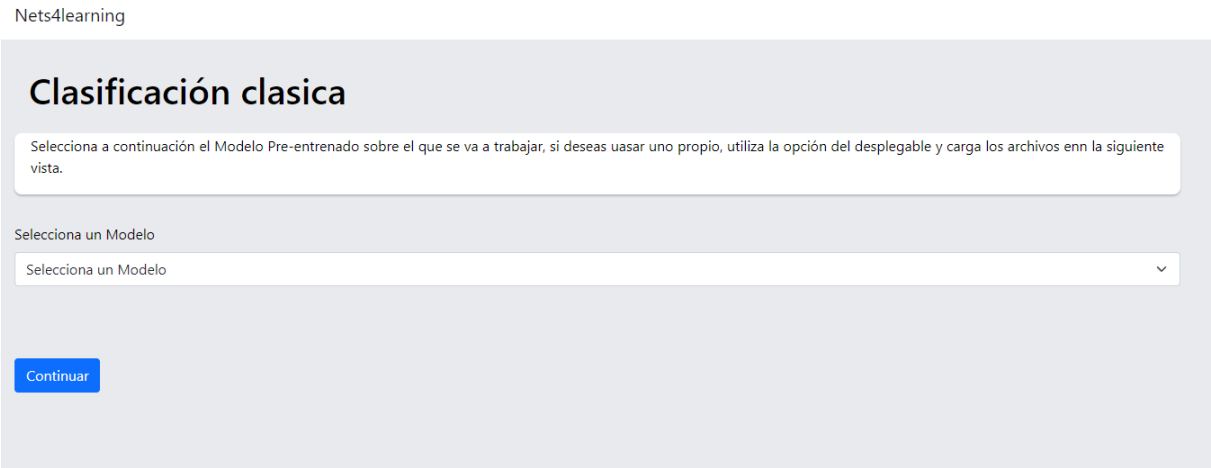

Figura 41: Interfaz selección modelo pre-entrenado

Esta interfaz nos permite seleccionar un modelo que ya ha sido entrenado para evaluarlo posteriormente, para ello simplemente debemos de seleccionar un modelo desde el desplegable y pulsar continuar.

Cabe destacar que una de las opciones del desplegable es “SUBIR MODELO PROPIO” (Figura 42).

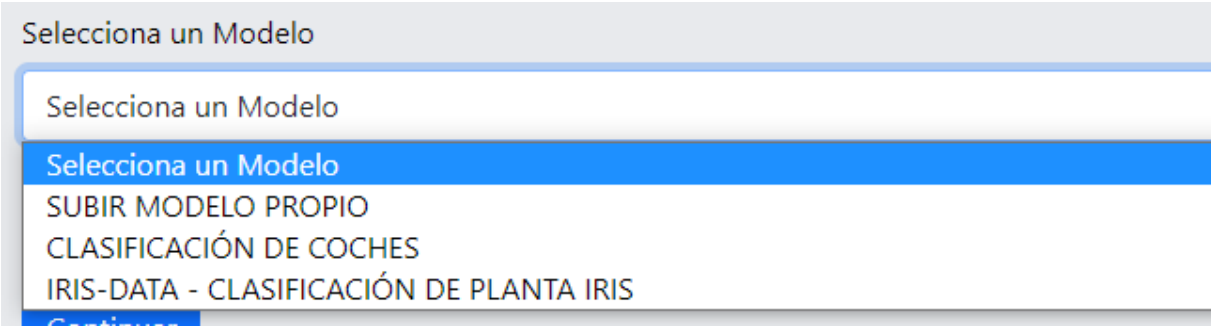

Figura 42: opciones desplegables correspondiente a los diferentes modelos a elegir de la Clasificación Clásica

Si seleccionamos esta opción en la siguiente vista tendremos un formulario para subir los archivos necesarios.

8.2.3. Crear/Editar arquitectura

De forma similar a la anterior obtenemos una interfaz que nos permite seleccionar un data set sobre el que se va a construir el modelo o arquitectura (Figura 43).

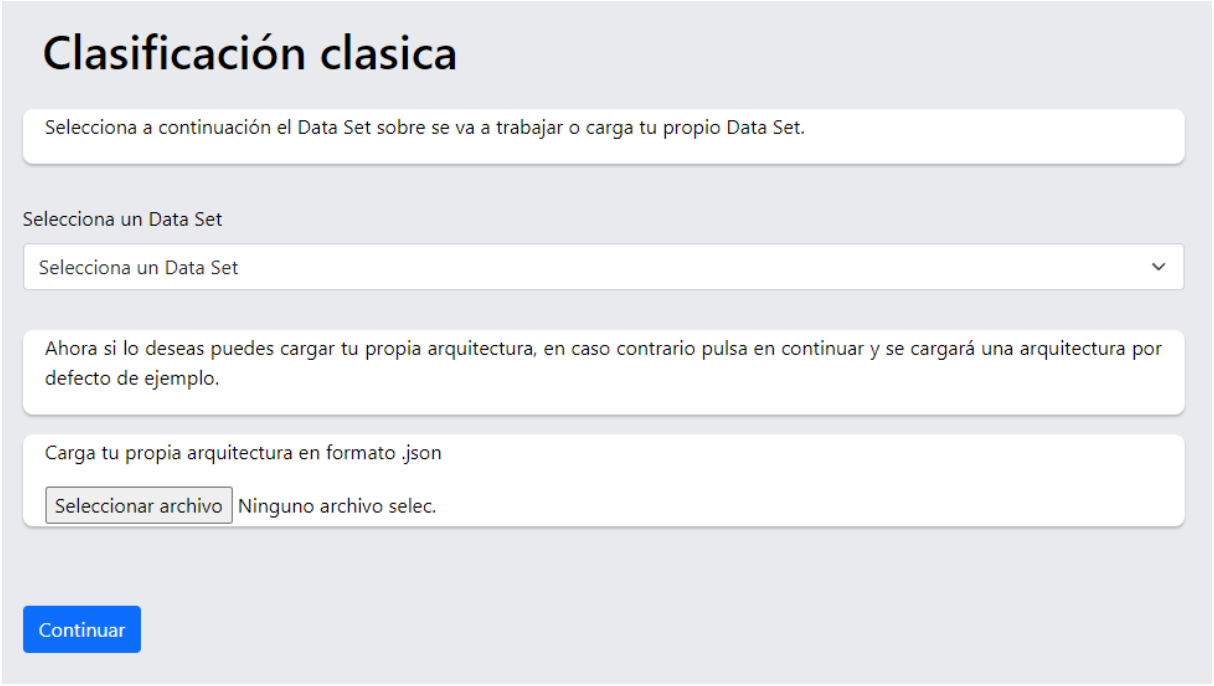

Figura 43: Interfaz tras seleccionar el botón "Crear/Editar arquitectura"

Este dataset lo podemos seleccionar mediante un desplegable que cuenta con la opción de "SUBIR PROPIO DATASET" (Figura 44), de forma que en la siguiente vista tendremos un formulario de cómo subir estos archivos.

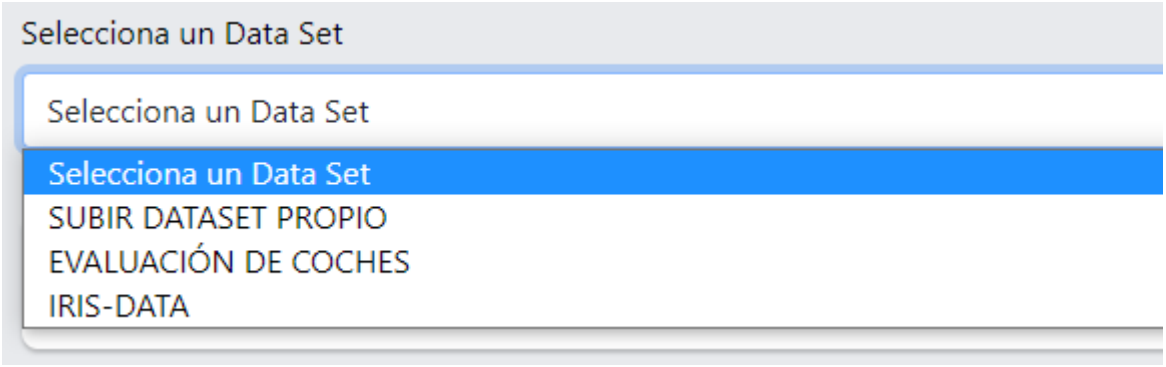

Figura 44: Desplegable de data sets

Además, tenemos la opción de subir nuestra propia arquitectura a partir de un archivo en formato .json.

Una vez hemos seleccionado el dataset y subido o no una arquitectura pulsamos el botón continuar.

8.2.4. Evaluación de modelos pre-entrenados

Si continuamos desde el punto 8.2.2 obtendremos la siguiente vista Figura 45 y Figura 46.

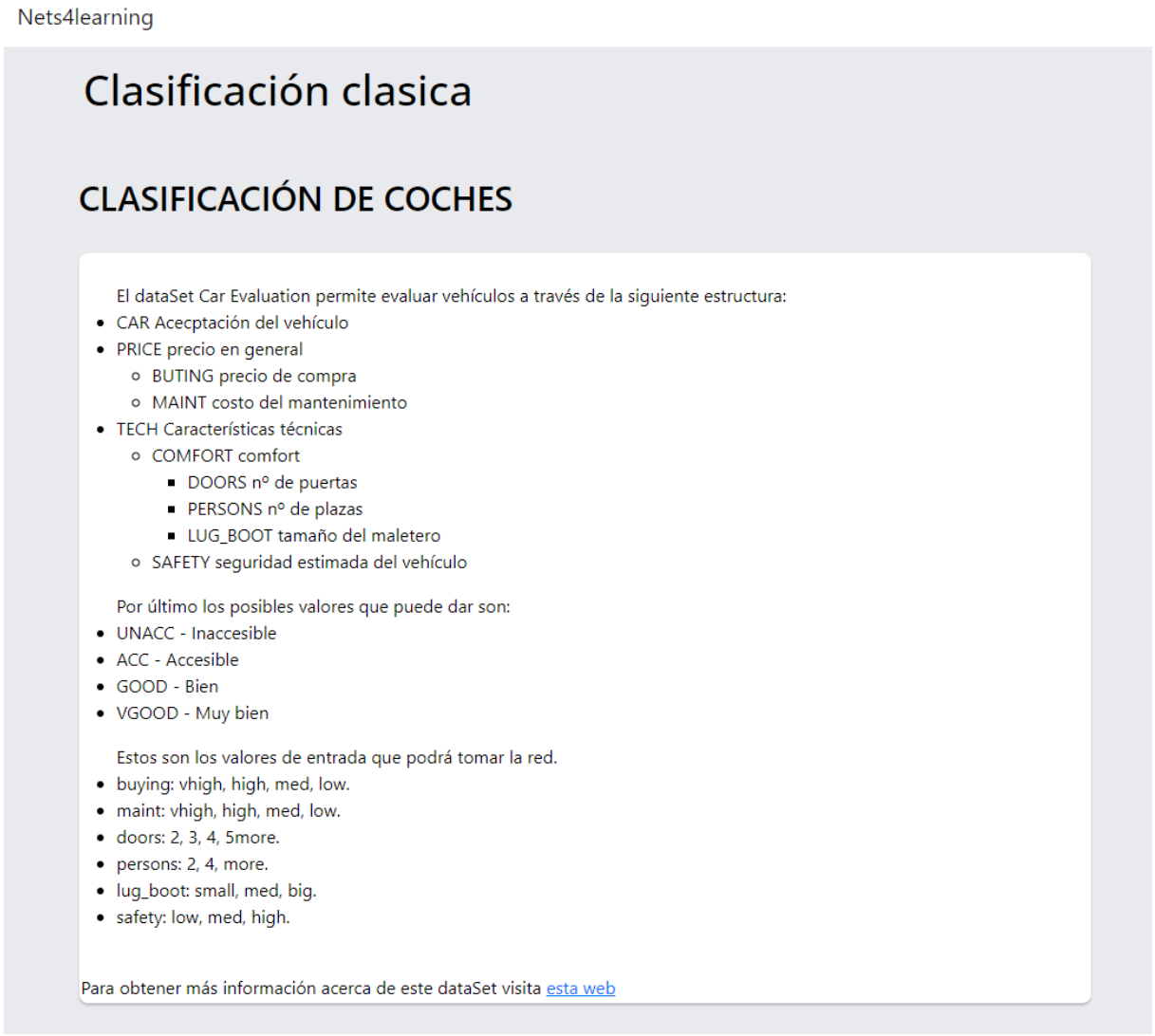

Figura 45: Interfaz evaluación modelos pre-entrenados (I)

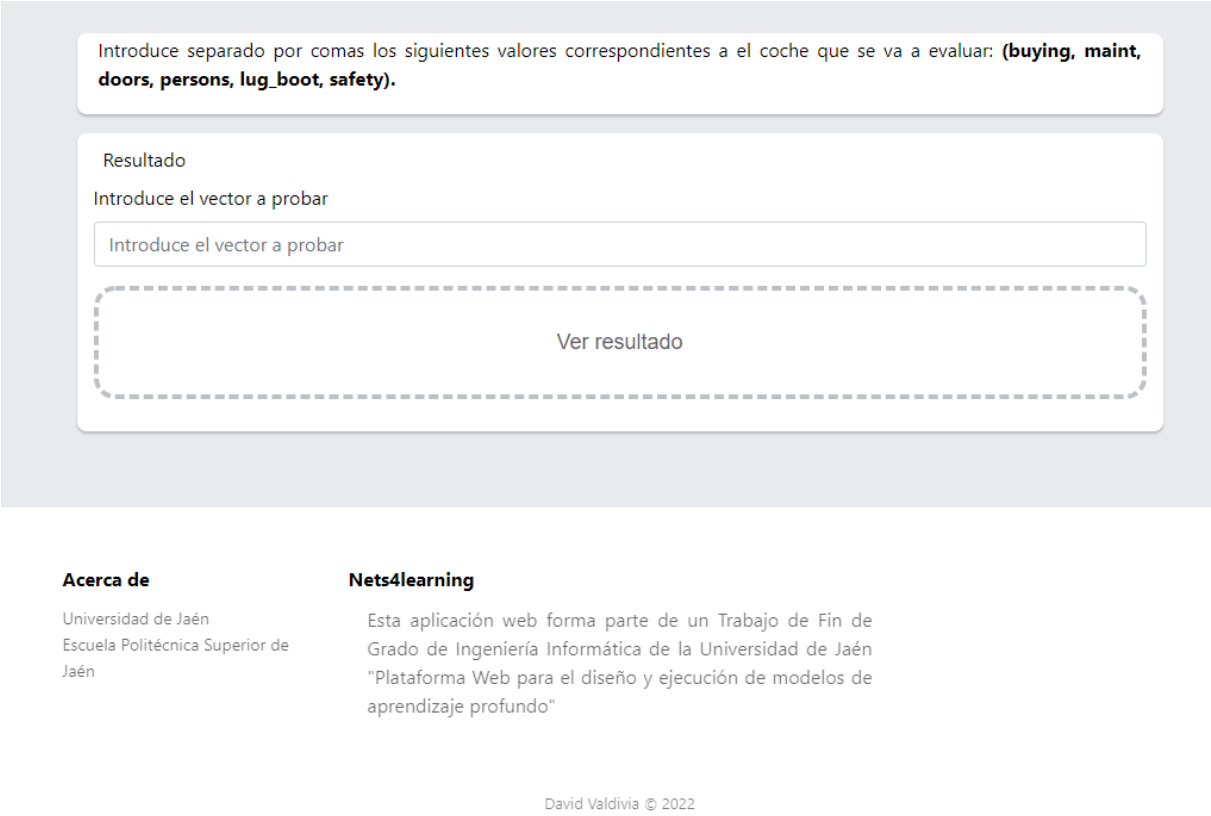

Figura 46: Interfaz evaluación modelos pre-entrenados (II)

Se puede observar que al principio tenemos una descripción del dataSet seleccionado, con un enlace donde se puede consultar más información del mismo.

Después tenemos un bocadillo de ayuda que nos informa como debemos de pasarle los parámetros al formulario para que la evaluación sea satisfactoria.

En último lugar nos encontramos con el formulario de evaluación, este varía dependiendo el tipo de disciplina que hayamos seleccionado al inicio. Hay tres tipos de entradas

- Entrada de texto plano en la que introducimos los diferentes valores separados por comas (Figura 46).
- Uso de webcam o formulario para subir una imagen (Figura 47).
- Formulario para subir una imagen (Figura 48).

Figura 47: Uso de webcam o fichero local como método de entrada para evaluar un modelo

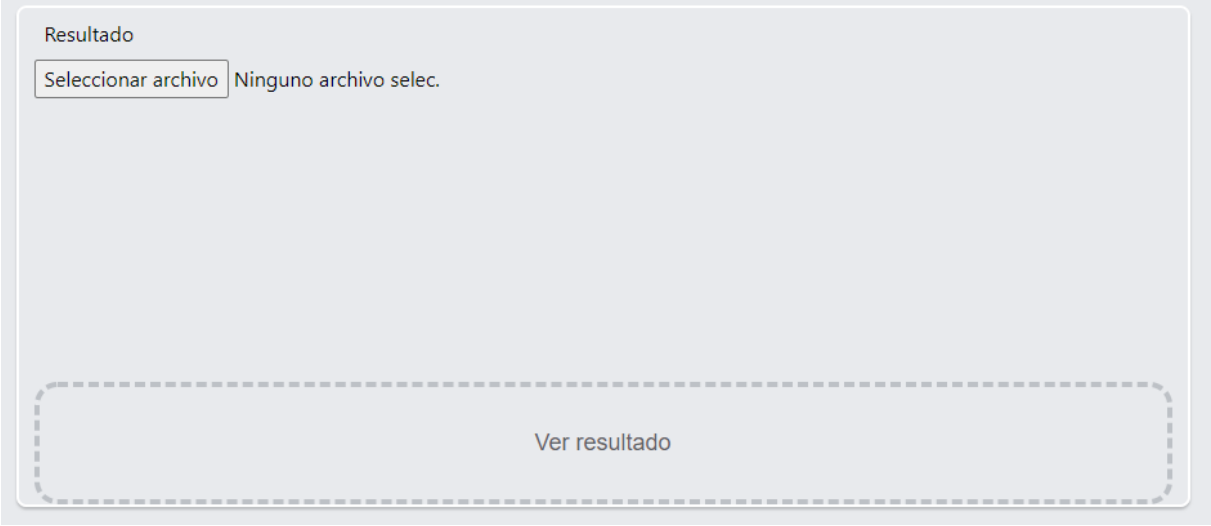

Figura 48: Carga de un fichero local como método de entrada para evaluar un modelo

En el caso de que anteriormente se haya seleccionado “SUBIR MODELO PROPIO” veremos al inicio un formulario para subir los archivos necesarios (Figura 49)

Figura 49: Interfaz para subir modelo propio

8.2.5. Creación y edición de arquitecturas

De igual forma que en el apartado anterior, al inicio tenemos una descripción del dataSet seleccionado. A continuación, se muestra una breve guía de cómo funciona la interfaz de usuario (Figura 50).

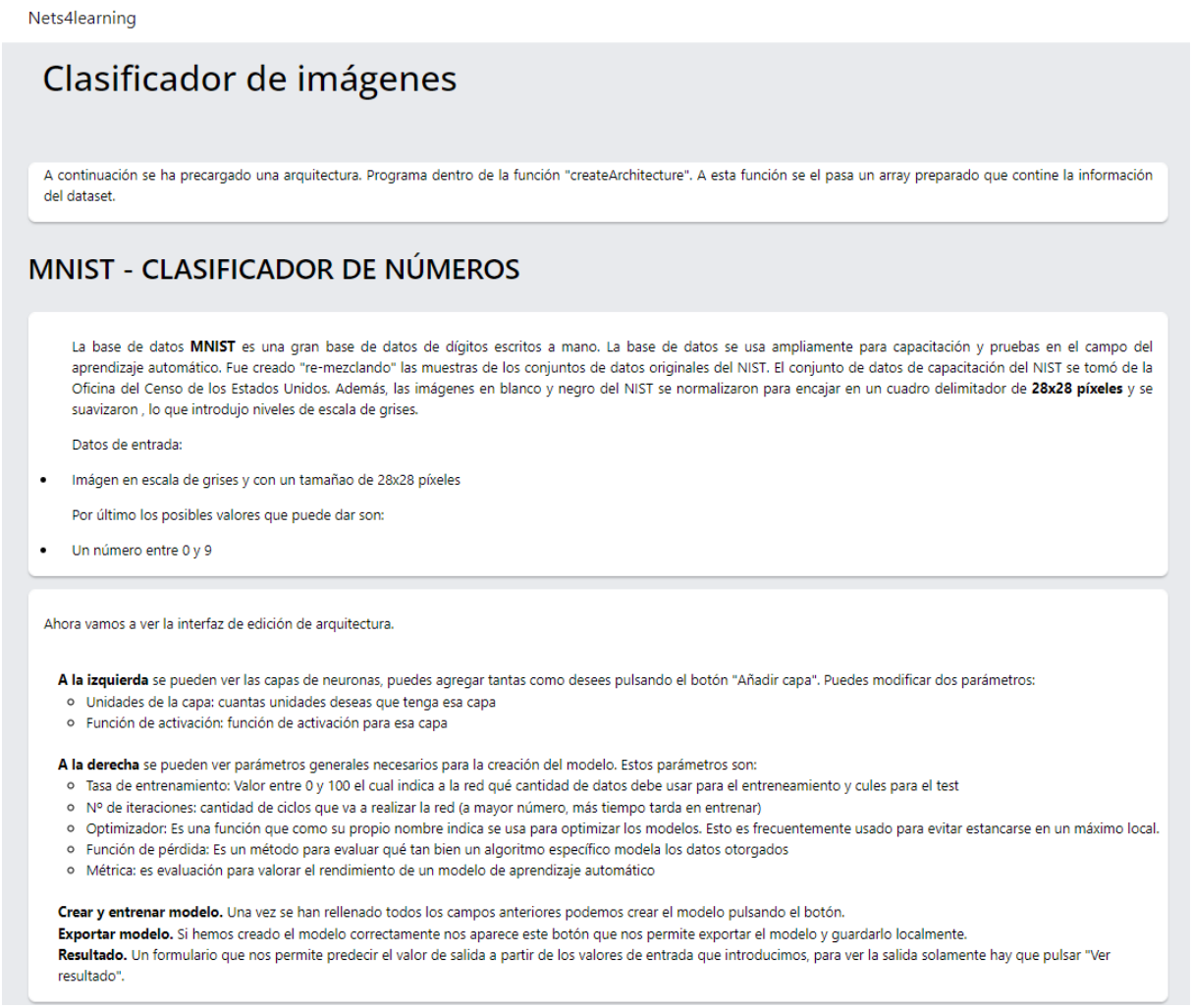

Figura 50: Interfaz creación y edición de arquitecturas (I)

Tras ese pequeño tutorial hay un diagrama de grafos que representan cada una de las capas que forman la arquitectura de nuestro modelo. Gracias a este gráfico se ve de una forma más intuitiva que se está haciendo en cada momento (Figura 51).

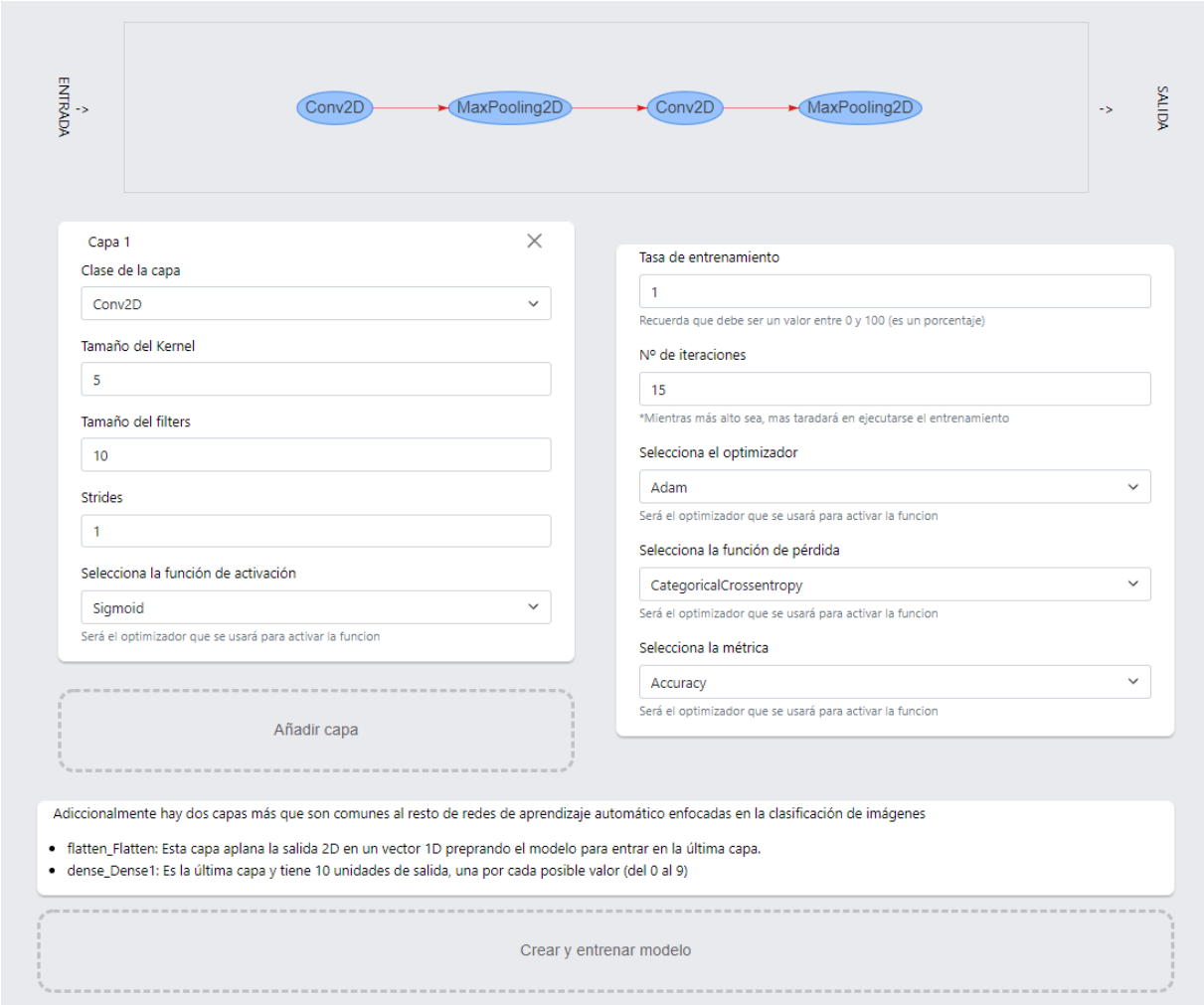

Figura 51: Interfaz creación y edición de arquitecturas (II)

Después hay un bocadillo a modo de información adicional, solo aparece si la categoría seleccionada lo requiere. Este bocadillo informa si la aplicación añade unas capas más después de las añadidas por el usuario, ya que son imprescindibles para el modelo.

Una vez se han modificado los datos de la arquitectura, podemos pulsar el botón “Crear y entrenar modelo” para, como el propio botón indica, crear el modelo a partir de los parámetros introducidos. Cuando pulsamos el botón nos aparecerá a la derecha una serie de gráficas que representan como va evolucionando el entrenamiento (Figura 52). Por comodidad este panel lateral se puede mostrar y ocultar pulsando la letra “ñ” de nuestro teclado.

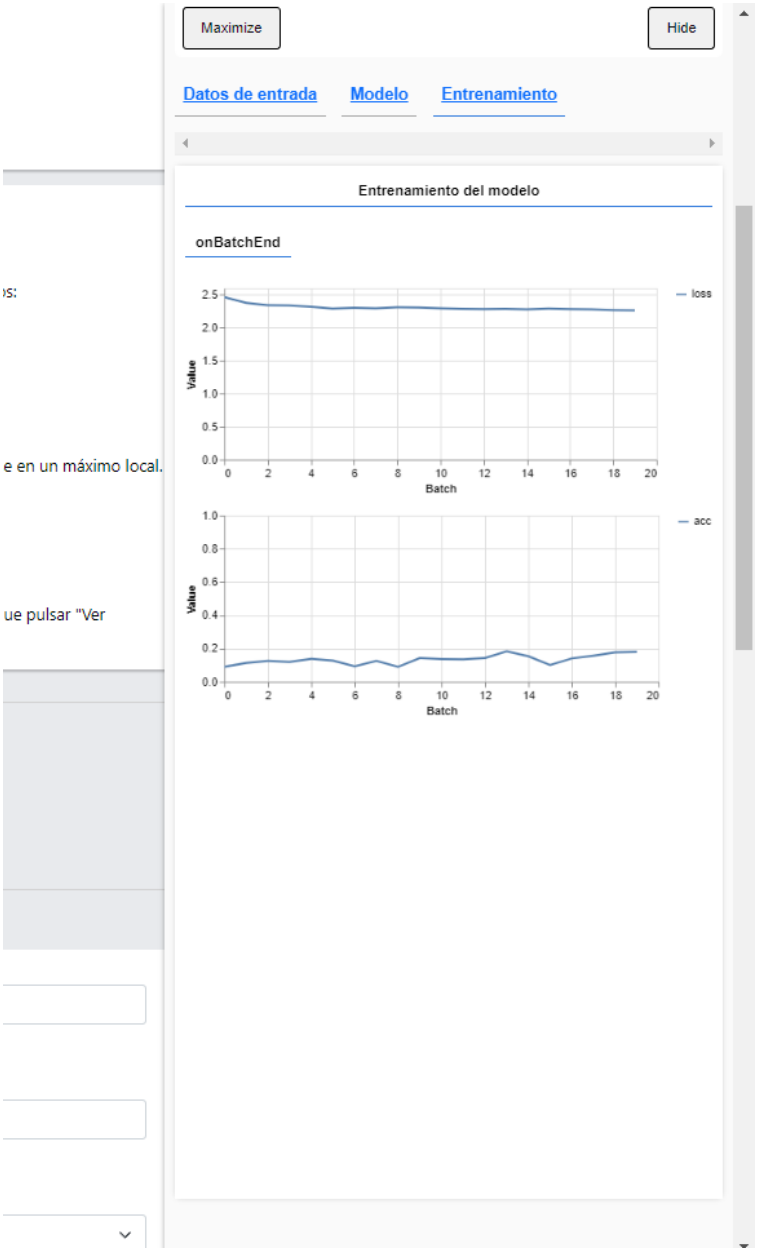

Figura 52: Panel lateral auxiliar al entrenamiento del modelo

Por último, tenemos el apartado de evaluación del modelo (Figura 53). En este caso nos encontramos con un canvas para dibujar y la opción de subir un fichero. Además, para otros ejemplos está el formulario de ingesta como se ha comentado en otro apartado.

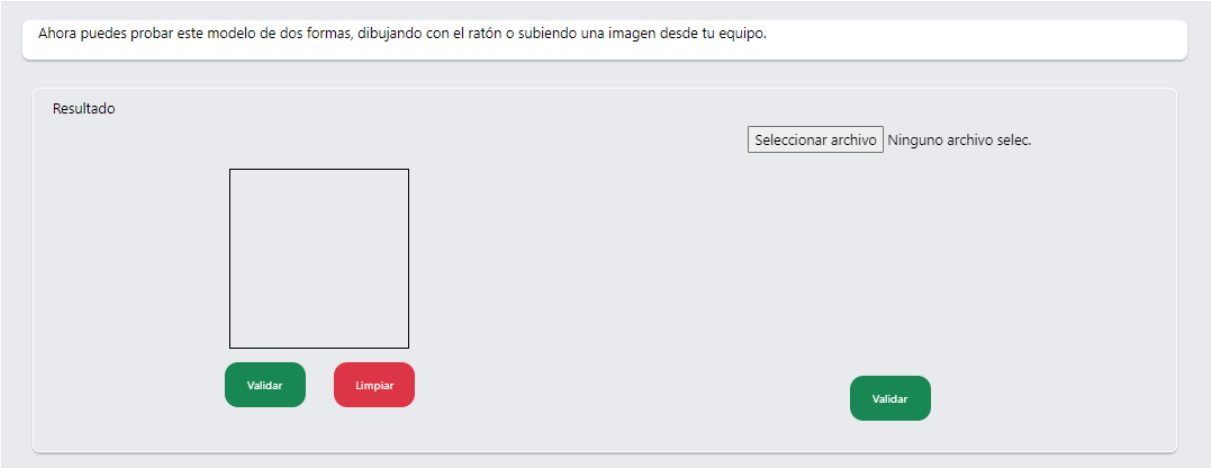

Figura 53: Interfaz creación y edición de arquitecturas (III)

8.2.6. Ejemplos de ejecución.

A continuación, se muestran unas capturas de la aplicación haciendo algunas predicciones.

En la Figura 54 se observa la respuesta de la predicción realizada por uno de los modelos preparados para la clasificación de imágenes.

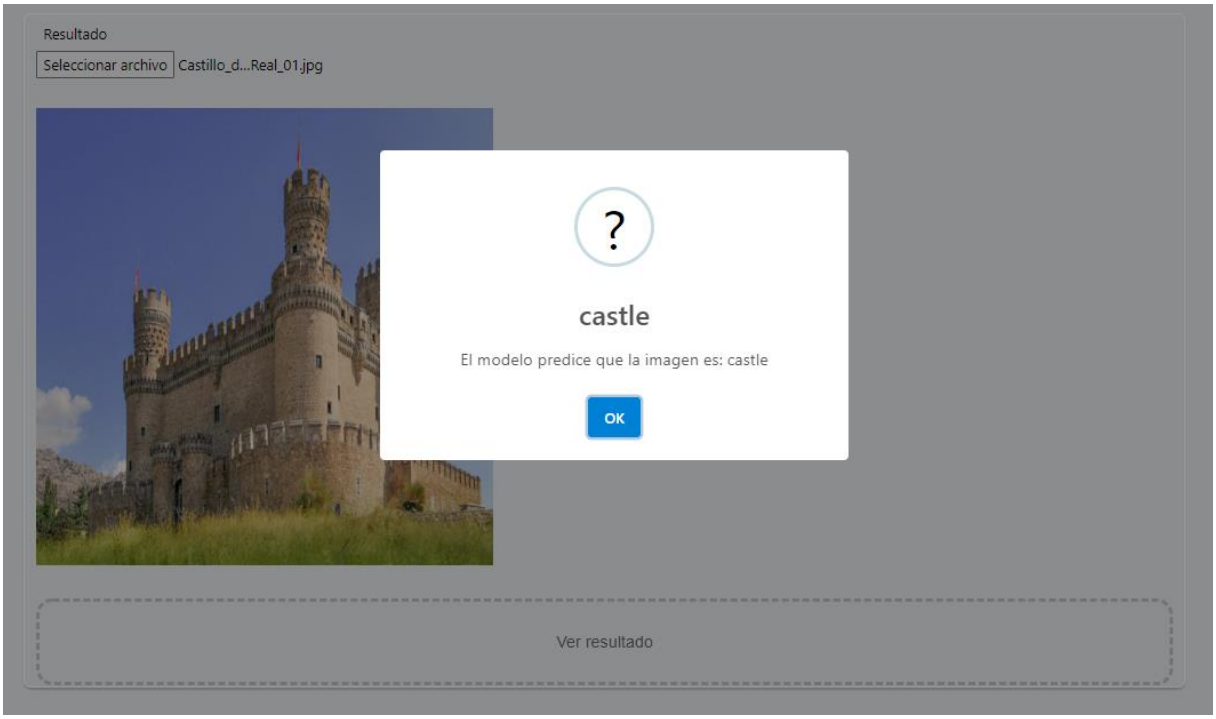

Figura 54: Evaluación de una imagen.

En la Figura 55 podemos ver dos imágenes, la imagen de la izquierda corresponde con la imagen original y la imagen de la derecha se corresponde con la predicción realizada por la red. Esta predicción ha sido realizada por el modelo Face Mesh de la categoría Identificación de objetos.

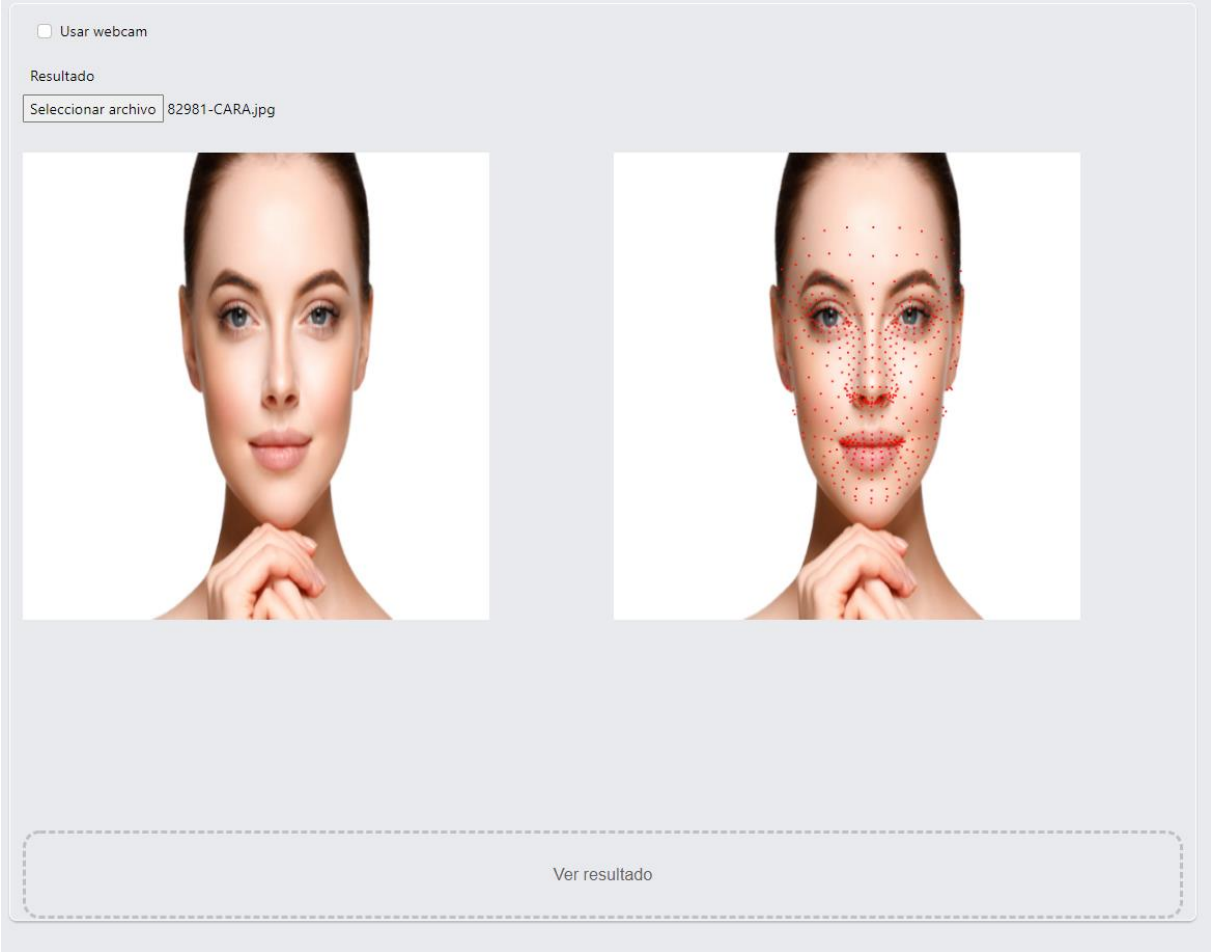

Figura 55: Evaluación del modelo Face Mesh sobre una imagen.

En la Figura 56 se observan dos imágenes de una persona. A la izquierda está la imagen original y en la derecha esta la imagen correspondiente con la predicción del modelo detector de articulaciones.

Figura 56: Evaluación del modelo detector de articulaciones sobre una imagen.

En la Figura 57 se observa la predicción del modelo Iris-data a través de un formulario. En este caso cada valor separado por punto y coma se corresponde con la longitud y anchura del pétalo y sépalo respectivamente.

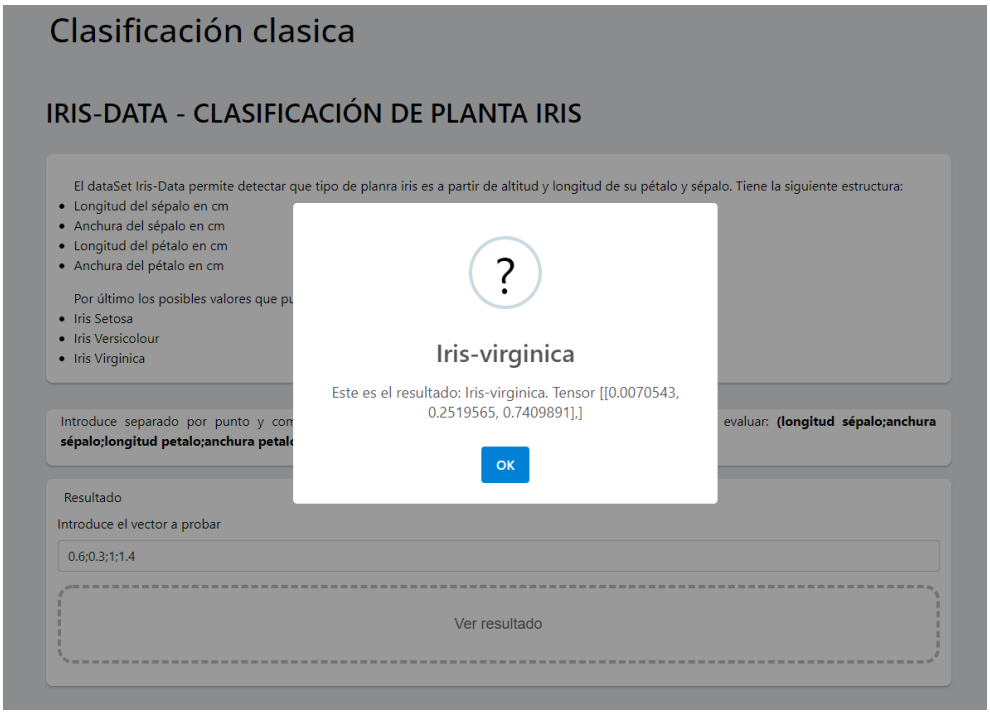

Figura 57: Evaluación del modelo Iris-data.

9. DEFINICIONES Y ABREVIATURAS.

9.1. Siglas.

TFG Trabajo fin de grado.

TDD Desarrollo orientado a pruebas.

DRA Desarrollo rápido de aplicaciones.

SSOO Sistemas Operativos.

.JSON JavaScript Object Notation, (notación de objeto de JavaScript) Extensión que corresponde con un formato de texto para el intercambio de datos.

.BIN Extensión correspondiente con un archivo binario.

CSS Hojas de estilo en cascada.

.MSI Extensión correspondiente a un archivo del paquete de instalación de Windows.

9.2. Referencias

Acervolima. (s.f.). Obtenido de https://es.acervolima.com/

Acervolima. (s.f.). *Funciones Tensorflow.js*. Obtenido de https://es.acervolima.com/funcion-tensorflow-js-tf-metrics-categoricalcrossentropy/

Bendersky, E. (18 de 10 de 2016). *The Softmax function and its derivative*. Obtenido de https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/

Boletín oficial del estado. (22 de 2 de 2018). *Convenio Estatal de las TIC.* Obtenido de https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf

Drew. (3 de 12 de 2019). *Ventajas y desventajas de la metodología Scrum.* Obtenido de https://blog.wearedrew.co/ventajas-y-desventajas-de-la-metodologia-scrum

España, R. (29 de 10 de 2020). *Qué son regresión y clasificación en Machine Learning*. Obtenido de https://agenciab12.com/noticia/que-son-regresion-clasificacion-machine-learning#:~:text=El%20an%C3%A1lisis%20de%20regresi%C3%B3n%20es,com%C3%BAn%20es%20la%20regresi%C3%B3n%20lineal

Estado, J. d. (5 de 12 de 2018). *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos.* Obtenido de https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.*

Google. (5 de 11 de 2021). *Modelos y capas*. Obtenido de https://www.tensorflow.org/js/guide/models_and_layers?hl=es-419

Grifol, D. (s.f.). *Metodologías de desarrollo ágil: TDD.* Obtenido de https://danielgrifol.es/metodologias-de-desarrollo-agil-tdd/#:~:text=TDD%20son%20las%20siglas%20en,para%20obtener%20el%20resultado%20deseado

IBM. (17 de 8 de 2021). *El modelo de redes neuronales*. Obtenido de https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

Instituto Tecnologico Autónomo de México. (2017). *Un sistema de recomendación basado en factorización de matrices y descenso en gradiente estocástico.* Obtenido de https://mariobecerra.github.io/files/school_projects/tesis_lma.pdf

IONOS. (17 de 8 de 2020). *Modelo en espiral: el modelo para la gestión de riesgos en el desarrollo de software.* Obtenido de https://www.ionos.es/startupguide/productividad/modelo-en-espiral/

Mas, D. (27 de 4 de 2019). *Metodología Kanban: pros y contras en la gestión de proyectos.* Obtenido de https://www.fhios.es/metodologia-kanban-pros-y-contras/

Microsoft. (14 de 4 de 2022). *Choose Between Traditional Web Apps and Single Page Apps (SPAs).* Obtenido de https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/choose-between-traditional-web-and-single-page-apps

Rennie, J. D., & Srebro, N. (2005). *Loss Functions for Preference Levels:.* Obtenido de https://home.ttic.edu/~nati/Publications/RennieSrebroIJCAI05.pdf

Russell, S. J., & Norvig, P. (2004). *Inteligencia artificial: un enfoque moderno*. Obtenido de https://books.google.es/books?id=yZCVPwAACAAJ

Saxena, S. (3 de 4 de 2021). *Binary Cross Entropy/Log Loss for Binary Classification*. Obtenido de https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/

Schmidhuber, J. (2015). *Deep Learning in Neural Networks: An Overview.*

Shen, Y. (2005). *Loss Functions For Binary Classification and Class Probability Estimation.*

Torres, J. (2020). *Python Deep Learning: Introducción práctica con Keras y TensorFlow 2.*

Velasco, L. (26 de 4 de 2020). *Optimizadores en redes neuronales profundas: un enfoque práctico.* Obtenido de https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-pr%C3%A1ctico-819b39a3eb5